2002

Scalable approaches for DiffServ multicasting

Aaron David Striegel
Iowa State University

Scalable approaches for DiffServ multicasting

by

Aaron David Striegel

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Manimaran Govindarasu, Major Professor
Arun Somani
Ahmed Kamal
Doug Jacobson
Soma Chaudhuri

Iowa State University

Ames, Iowa

2002

UMI Number: 3073480

UMI Microform 3073480

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Aaron David Striegel

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

Acronym	Description
AF	Assured Forwarding
BB	Bandwidth Broker
BE	Best Effort
DS	DiffServ
DSCP	DiffServ Code Point
DSMCast	DiffServ MultiCasting
EBM	Edge-Based Multicasting
ECN	Expedited Congestion Notification
EF	Expedited Forwarding
EI-HELLO	Edge-based Intelligent HELLO
HBI	Heartbeat Interval
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSec	IP Security header
ISP	Internet Service Provider
MB	Multicast Broker
NSIS	Next Step in Internet Signaling
PHB	Per-Hop Behavior
PIM	Protocol Independent Multicast
QoS	Quality of Service
RSVP	Resource ReSeVation Protocol
SGM	Small Group Multicasting
SLA	Service Level Agreement
SM	Sparse Mode
SSM	Source-Specific Multicast
TEH	Tree Encapsulation Header
TTL	Time To Live
VoIP	Voice over IP

ABSTRACT

Over the last several years, there has been an explosion in the introduction of new Internet technologies. Whereas the Internet in its original form was a medium primarily for academia and research interests, the Internet has been redefined as business and consumer interests have dominated the focal points of Internet technology. The dominant question facing the Internet today is, how can the network meet the needs of the users and their applications while trying to keep such implementations scalable to the billions of users present on the Internet? Two of the emerging technologies for answering the question are Differentiated Services (DiffServ) and multicasting. Although the two technologies share complementary goals, the integration of the two technologies is a non-trivial issue due to three fundamental conflicts. The three fundamental conflicts are the scalability of per-group state information, sender versus receiver-driven QoS, and resource management. The issues surrounding how to solve these conflicts provide the basis for this dissertation.

In this dissertation, two architectures (DiffServ Multicasting (DSMCast) and Edge-Based Multicasting (EBM)) are proposed to satisfy the requirements for scalable DiffServ multicasting architectures. In addition to the two architectures, this dissertation also presents the first in-depth study regarding single tree support for heterogeneous QoS multicasting. Furthermore, the dissertation proposes a novel application of DSMCast for fault tolerance and management of the DiffServ network. Finally, the dissertation comments on future applications of the architectures and proposes several areas for future research.

CHAPTER 1. INTRODUCTION

Over the last several years, there has been an explosion in the introduction of new Internet technologies. Whereas the Internet in its original form was a medium primarily for academia and research interests, the Internet has been redefined as business and consumer interests have dominated the focal points of Internet technology. As a result, new applications are continually emerging that require vastly different characteristics of the underlying network versus the original applications of the Internet. Examples of such applications include video-conferencing, peer-to-peer (p2p) file-sharing [1], large scale content distribution [2, 3], distributed computing, and on-line gaming.

1.1 Evolution of the Internet

Although the original Internet architecture provides sufficient functionality for basic text and file transfer applications, the architecture is insufficient when faced with the requirements of these new applications. The fundamental reason why the original Internet architecture is insufficient can be directly attributed to two key trends: a new demand for quality of service (QoS) and a shift in the focus and scale of the applications.

Unlike today, the original driving force of the Internet was basic connectivity itself. Since any connectivity was an infinite improvement over zero connectivity, the issue of the quality of such connectivity was not a primary consideration. In addition, the notion of connectivity was driven by the paradigm of two computers exchanging information with one another. The Internet itself arose out of the medium that provided such connectivity. As a result, many of the original Internet applications (e-mail, FTP) focused on the basic function of simply operating together rather than the quality of such operations. This focus was fueled in part by the fact

that the original Internet represented an unprecedented mechanism for exchanging information. Even at a relatively low quality, the Internet offered information transport on a scope and speed that far surpassed anything previously available. Recognizing that such connectivity could offer tremendous benefits to productivity and leisure activities, businesses and consumers drove the Internet explosion of the 1990's. The founding principle of connectivity served the Internet well as new users were able to become part of the Internet in quite short order.

1.1.1 From Connectivity to QoS and Beyond

However, once the primary connectivity issues were resolved, the applications began to evolve from demanding connectivity to demanding requirements for interactivity and hence QoS. Rather than the delivery of information being sufficient, applications demanded requirements on when and how information would be delivered. For instance, no longer would applications be satisfied with packets simply being delivered some time in the future, packets needed to be delivered within a predictable delay or delivered with a predictable loss. The interactive and multimedia nature of the new applications represented a fundamental shift whereby the utility of the application became directly tied to the quality of the information transport.

In addition, the scale and focus of the applications began to shift as well. Whereas significant improvements in interactivity such as the World Wide Web (WWW) piqued the general interest of the public, the interactivity was still focused on the single user or point-to-point experience. The next natural step was a shift in the focus of such applications to evolve from single user applications to group-oriented environments. The new group-based applications depend on not only QoS requirements from the network but also on network services on a scale and scope well beyond the scope of the original Internet. These group-based applications span across tens, thousands, or even millions of users, all demanding QoS local to themselves. Not only do these applications seek to unite large numbers of heterogeneous users but also the number of applications and their instantiations has increased as well.

The underlying network is faced with the difficult task of serving an ever increasing pool of users and applications, all with possibly conflicting QoS requirements. Although the architects

of the original Internet took into account the issues associated with scalable connectivity (i.e. how to offer point-to-point connectivity to many users), the myriad of issues surrounding QoS and the trend towards group applications were not a focal point.

1.2 Problem Motivation

Thus, the dominant question facing the Internet today is, how can the network meet the needs of the users and their applications (QoS) while trying to keep such implementations scalable to the billions of users present on the Internet? Two of the emerging technologies for answering this question are Differentiated Services (DiffServ) and multicasting. DiffServ proposes a model for QoS while multicasting proposes a model for group applications. Although the two technologies share complementary goals, the integration of the two technologies is a non-trivial issue. The issues surrounding how to integrate these two technologies provide the basis for this dissertation.

The remainder of this dissertation is organized as follows. The remainder of Chapter 1 outlines the background material regarding quality of service (QoS), Differentiated Services (DiffServ), and multicasting. Next, Chapter 2 details the problems faced when trying to integrate DiffServ and multicasting. The chapter categorizes the different approaches to the problem and discusses the previous work on the subject.

Chapter 3 presents the first architecture for DiffServ multicasting, DSMCast (DiffServ Multicast). The chapter proposes the DSMCast architecture, discusses the tradeoffs, and analyzes the performance of the DSMCast architecture. Following the DSMCast architecture, Chapter 4 presents the next architecture, EBM (Edge-Based Multicasting). Similar to Chapter 3, the chapter discusses the approach, the tradeoffs, and presents simulation studies comparing EBM and DSMCast as well as other approaches.

Next, Chapter 5 addresses the issue of heterogenous QoS in DiffServ multicasting. The chapter presents the basis for the problem and presents interesting facets of the problem based on how various approaches attempt to solve the problem (including the architectures posed by this dissertation). Chapter 6 presents a unique application of DSMCast for expediting the

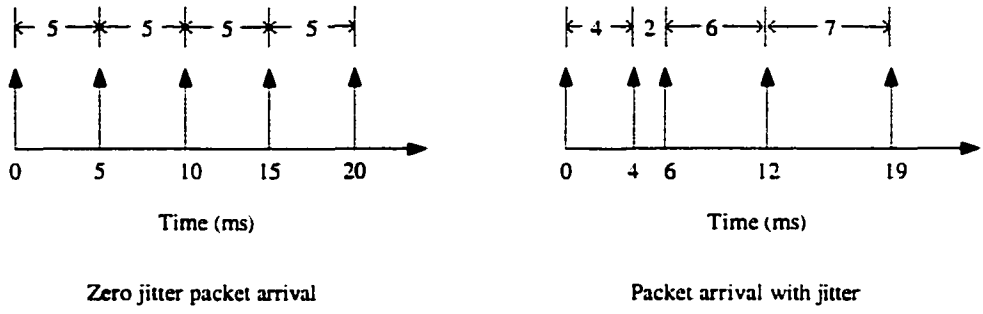


Figure 1.1 Delay jitter in packet arrival

fault detection of the underlying link state protocol named EI-HELLO (Edge-based Intelligent HELLO). The chapter discusses the underlying premises of EI-HELLO and examines its performance improvements through simulation. Finally, Chapter 7 concludes the dissertation by addressing other applicable issues to the dissertation, offering several concluding remarks, and discussing future research directions.

1.3 What is QoS?

One of the most basic questions is to define what QoS actually encompasses. Traditionally, QoS has encompassed two main parameters, the loss rate and the end-to-end delay that packets receive as they traverse the network. In their simplest form, these two parameters provide the foundation for defining QoS in the network.

1.3.1 Beyond Loss and Delay

In actuality, QoS encompasses much more than the two parameters of loss and delay. Although loss and delay have a profound effect on the QoS as perceived by the user, many other parameters can also affect QoS. Several of these parameters include:

- *Delay jitter:* Ideally, packets arrive in a predictable manner to the user. However, the user may experience delay jitter when packets arrive with inconsistent delays (see Figure 1.1).

Packets that arrives early may need to be buffered, thus requiring additional resources at the user that may or may not be available.

- *Loss window:* Although the packet loss experience by an application may be sufficient on average, the locations at which the losses occur can have a significant effect on the QoS perceived by the user. For instance, in Voice over IP (VoIP), a loss of several consecutive packets in a row may cut out an entire talkspurt whereas the same loss rate distributed over time may be imperceptible to the user [4, 5, 6].

- *Fault tolerance:* For extremely critical communications, it may be not only a requirement to have zero loss but also that loss cannot be tolerated at any time, regardless of faults in the network. Examples of such applications might include remote surgery or process control. In such cases, it may be necessary to devote additional backup resources to meet the necessary levels of redundancy [7].

- *Security:* Depending upon the sensitivity of the information transmitted by the application, it may be necessary to employ security to the packets traversing the network. Security measures may range from *authentication* [8], verifying the identity of the sender, to *authorization*, verifying the user can use the resources, to *encryption*, hiding the contents of the packet [9].

1.3.2 QoS Parameter Considerations

One of the first considerations regarding QoS is the tolerances of applications regarding their QoS demands. To start, is a fixed (bounded) QoS necessary or will an average QoS suffice? Is the QoS violation critical to the operation of the application or can occasional violations be tolerated? For example, a handheld device may have an absolute threshold for delay jitter due to memory requirements but a desktop machine may be able to tolerate an average delay jitter. Another example might a bounded delay for a VoIP phone call but a sliding delay for a user watching streaming video. Several examples of works in these areas include the absolute QoS work done by the IETF [10, 11] and several new works based on

variable but predictable QoS [12, 13, 14, 15].

A second consideration is how the QoS is conveyed to the network. Does the user push their QoS to the sender? Alternatively, does the sender push QoS to the user? Although the user is ultimately the one that is the recipient of the QoS, various schemes may rely on either the sender or receiver defining the QoS levels of the network. For instance, a media provider may want to offer streaming video to users at different service levels. In one possibility, the user has a profile that defines the desired service at the media provider. The server chooses the QoS to meet the QoS desired by the user. Alternatively, the user defines the QoS and sets up the network resources for the connection(s). Both approaches have their merits and are discussed in more detail in Chapter 5.

A final consideration for QoS is how different QoS levels are billed. Should the cost be paid by the sender, by the receiver, or by both? For group communications, how does one divide the cost of the connection? A considerable body of work that is still ongoing has been devoted to this non-trivial topic [16, 17, 18]. On the one extreme, accurate billing is highly desirable but at the same time, one does not wish the cost of the billing scheme to outweigh the benefits it introduces.

1.3.3 Delivering QoS

In order to meet the basic QoS requirements of the applications using the network, there are two primary schools of thought governing how resources should be allocated. The first school of thought is to increase the bandwidth available to users such that the extra capacity of the network allows all users to meet their appropriate QoS. By providing capacity beyond the needs of the users on the network (over-provisioning), the network will not become congested and thus all users will be able to meet their QoS. QoS is provided by default and requires no additional overhead in the routers or by the end-users themselves.

In contrast, the second school of thought is that bandwidth can never be considered unlimited and therefore the limited bandwidth should be appropriately prioritized among users. Although the bandwidth of the Internet backbone has dramatically increased [19], the back-

bone of the Internet itself is still far from being able to support QoS without appropriate resource allocation mechanisms.

In fact, the increase in bandwidth has had much of the opposite effect as new applications are being continually developed that erode gains in network capacity. Examples of such applications include many of the newly emerging group-based QoS applications such as on-line gaming, peer-to-peer computing [20], and distributed workspaces. The deployment of broadband capabilities to the end-user has only accelerated the demand for additional bandwidth as end-users are able to take advantage of these new applications. Thus, for the foreseeable future, some form of resource provisioning is necessary to provide QoS across the Internet.

1.4 Evolution of QoS in the Internet

Although the notion of priority has been present since the inception of the IPv4 protocol, the preferred method for supporting QoS has evolved over time. This evolution has occurred as not only the applications of the Internet have evolved but also as the scale of the Internet has evolved as well.

1.4.1 IPv4 ToS

The original concept for QoS was the ToS (Type of Service) field of the IPv4 protocol [21, 22]. Under this scheme, users could mark their packets to whatever service level that they desired. An application requiring a higher priority could mark its packets as high priority and hence receive priority scheduling from each of the routers along the way. Applications that did not demand higher levels of QoS would mark their packets appropriately.

The problem with this scheme is that it relied entirely on the users to specify their own QoS without any sort of policing or shaping mechanism. Thus, there was little incentive for a user not to mark packets at the highest priority and hence, the system broke down into a best-effort scheme whereby all packets were marked at the same priority level. The second problem was that the behavior associated with the ToS was not consistent between different domains. Whereas one domain could obey the ToS markings, another domain could simply

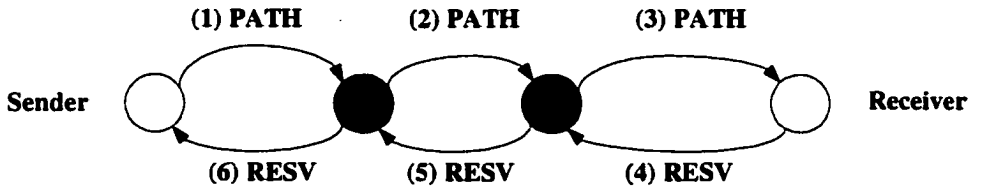


Figure 1.2 IntServ model using RSVP

ignore the ToS markings. Hence, the IETF (Internet Engineering Task Force) recognized the need for an end-to-end QoS scheme that would be implemented and deployed throughout the global Internet.

1.4.2 Integrated Services (IntServ)

The first model developed by the IETF for QoS was the Integrated Services model or IntServ [23]. The IntServ model aims to provide an absolute guarantee of QoS for each flow (connection) across the network. Each flow would reserve the resources that it required and flows would be policed at each router in the network. This model provides two broad categories of service, *guaranteed service* [24] for real-time flows requiring strict delay and jitter bounds, and *controlled load service* [25] that is designed to offer the QoS guarantees of a lightly loaded network for adaptive multimedia applications.

In order to send data on the network, an application must first reserve such resources on the network using a signaling protocol known as RSVP (Resource ReSerVation Protocol [26]). RSVP provides both resource reservation (allocation of resources to the individual flow) as well as admission control (determination if the resources exist in the network for the flow). Although IntServ is dependent upon RSVP, RSVP is not dependent upon IntServ and thus may applied to other models.

Once an application has successfully reserved resources on the network, the application may transmit the information onto the network (see Figure 1.2). At each router along the path of the connection, the packets are appropriately shaped and policed according to the initial RSVP reservation. The flow is identified by either a combination of both the transport and

networks layers or strictly by the network layer (ex. source address/flow ID tuple of IPv6). During the lifetime of the connection, the resources of the flow must be periodically refreshed to ensure that the resources stay allocated. This *soft state* is used to ensure that resources do not become permanently allocated in the event of an end system or router failure. When the flow is finished, the resources are released back to the network. Soft state also offers benefits when supporting group-based applications since the end user does not have to explicitly leave the group.

1.4.2.1 Strengths and Weaknesses

The primary strength of the IntServ model is that it provides an absolute service guarantee once a flow is admitted. As all flows are strictly policed, other flows cannot have an adverse impact on the QoS if they violate their respective allocation of resources. In addition, IntServ provides a receiver-driven QoS in that all resource reservations are driven by the receiver (see Figure 1.2). Thus, the receiver can select its desired resources based on its capabilities without the sender having to discern the capabilities of the receiver.

However, the IntServ model has several key weaknesses. First and foremost, each router requires a significant amount of processing overhead as each router is required to maintain state information for each flow. On the Internet, several million or even several billion flows may be flowing across a router, thus presenting scalability difficulties. In addition, IntServ is impractical for short-lived flows since the connection setup overhead is often greater than the transmission of all of the packets in a flow. For services such as HTTP (which currently dominates Internet traffic), the entire flow can consist of as little as four packets. Consequently, the Differentiated Services (DiffServ) model [27, 28] was proposed as an alternative model for providing QoS over the Internet that overcomes the shortcomings of the IntServ model.

1.4.3 Differentiated Services (DiffServ)

The primary goal of DiffServ was to provide the benefits of QoS without the scalability limitations of IntServ. Rather than attempting to deal with QoS on a per-flow basis, the

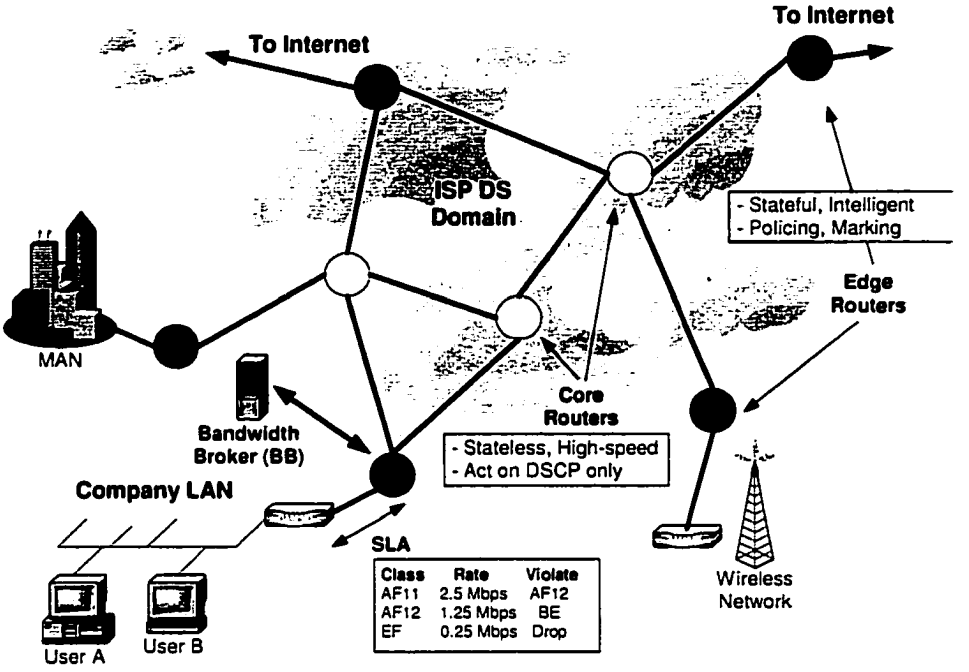


Figure 1.3 DiffServ model

DiffServ model aggregates traffic with similar QoS requirements into classes of traffic. DiffServ does not maintain per-flow information, thus eliminating the two key weaknesses of the IntServ model, the setup and the maintenance of per-flow state information. In addition, DiffServ focuses on domain-wise (AS) behavior rather than end-to-end behavior in order to expedite the deployment of DiffServ.

1.4.3.1 DiffServ Architecture

In the DiffServ architecture, intelligence is migrated to the edge of the domain in order to keep the core of the network simple and scalable. Routers in a DiffServ domain are divided into two categories, *core routers* (simple and high speed) and *edge routers* (stateful and intelligent). Core routers do not have per-flow state information and differentiate packets according to the marking of the packet. In contrast, edge routers are stateful entities that are responsible for policing and/or marking all packets according to an *SLA* (Service Level Agreement) between

the source (other ISP, user, company) and the domain, or between two domains. Figure 1.3 shows a picture of the DiffServ architecture.

When a packet enters the network, the packet must first pass through an edge router. The edge router is responsible for policing and shaping all packets that pass onto the DiffServ network. Without proper operation of the edge router, the network would decay into best effort service. If necessary, the edge router may also mark the packet if the source is not DiffServ aware. Alternatively, the edge router may re-mark or drop a packet in the event that the flow or source has violated its respective SLA.

The SLA that governs the policing of the respective flow may be either of a static or dynamic nature. If an entity known as a Bandwidth Broker (BB) is present in the domain, senders or receivers may negotiate for additional bandwidth or different packet treatments (shaping/PHBs) from their SLA and the network. The protocols to negotiate such resources are still under development by the IETF and research community. Several proposals have been introduced that include using RSVP [29], dynamic allocation [30], as well as others [31].

1.4.3.2 Packet Scheduling in the Core

Once a packet enters the network, the packet is scheduled for transmission on the link according to the marking (DSCP - DiffServ CodePoint) of the packet. In order to allow for gradual deployment, the DS (DiffServ) field is simply a semantic change to the original ToS field of IPv4 and the Flow ID field of IPv6, thus allowing DiffServ to operate without major changes to the Internet framework. The DS field is made up of six bits with the remaining two bits left over for protocols such as ECN (Expedited Congestion Notification [32]).

The DSCP of the packet defines the PHB or Per-Hop Behavior that should be applied to the packet for scheduling. The PHB defines how the packet will be scheduled as well as dealt with during congestion. In addition, certain PHBs may be rate-limited as well (such as Expedited Forwarding [10]) to prevent link starvation of lower classes by high priority traffic. Each core node in the network is independent of the other core nodes in order to keep scheduling simple and efficient. The underlying scheduling mechanism is left up to the manufacturer with various

options to choose from [14, 33, 34, 35, 36].

The IETF has defined several PHBs to be backward compatible with the original ToS markings. These PHBs include an Expedited Forwarding (EF) PHB [10] for low-loss, low-delay traffic such as VoIP. Another PHB is the Assured Forwarding (AF) PHB [11] which defines only the congestion priority of the traffic (low, medium, high). A best effort PHB has also been defined by the DiffServ working group. In addition, several other PHBs have also been proposed as IETF drafts [37].

1.4.3.3 Strengths and Weaknesses

The primary strength of the DiffServ model is that it provides QoS in a scalable fashion by aggregating traffic into classes rather than dealing with traffic on a per-flow basis. In addition, the stateless nature of DiffServ is highly beneficial to short-lived flows due to the absence of connection setup costs. As a result, simple, high-speed routers can be developed that can scale not only to the number of flows present in the network but also to the increasing speed of the underlying network links. In fact, the recent pace of advancement in network capacity is far surpassing the increases in computational performance [38], thus necessitating simple and effective solutions for QoS.

However, the DiffServ model does suffer from several weaknesses. First, the PHB-centric nature of DiffServ and the aggregation of traffic into classes provides an average (coarse-grained) QoS rather than a fine-grained QoS. Thus, although the average QoS packets in the class may satisfy the requested QoS, it is still possible for QoS violations of individual flows due to the aggregation of the underlying traffic.

Second, since the core does not police or shape the majority of traffic, incorrect configurations or malicious configurations of edge router policing or incorrect routing can have implications on the QoS provided on links in the core of the network. This problem would not occur in IntServ as each flow is policed at each and every router along the path, thus preventing one instance of incorrect policing/shaping from impacting other links in the networks.

Finally, the six bits of the DS field and its backward compatibility with the ToS markings

can potentially limit the number of unique QoS treatments that can be applied across a given domain. However, this is a relatively minor point as IPv6 can offer a up to thirty bits for the DS field. In addition, the inclusion of six bits for unique classes (64 different classes) represents a significant improvement over the current best-effort (single class) nature of the Internet.

1.4.3.4 Beyond DiffServ

Although there has been significant work by both the research community and the IETF, there are still several research questions that need to be answered. Several of these questions include:

- *Multicasting:* During the initial work of the DiffServ working group, the issues regarding DiffServ multicast were raised through several IETF drafts [39, 40] but the final work was left to other IETF groups. As mentioned earlier, it is this gap that provides the motivation for this dissertation and such issues are discussed in more detail in Chapter 2.

- *Inter-domain signaling:* Although there has been some work in the research community regarding inter-domain resource reservation and signalling, the research surrounding these issues is still in its initial stages [41, 42, 29]. The DiffServ working group has deferred such actions to the IETF NSIS working group is actively pursuing solutions that encompass not only DiffServ but also future QoS models for the Internet [43, 31].

- *QoS Granularity:* The aggregation effect of DiffServ can have the potential for significant variations of the QoS of individual flows within a class. Although the QoS of the class may be satisfied, the QoS of individual flows within a class can vary significantly. The topic is a relatively unexplored research area that merits future attention [44, 45].

- *Provisioning:* As the initial focus of DiffServ was on per-hop behavior and not necessarily end-to-end performance, the next natural step is to deliver predictable edge-to-edge differentiation. The issue of provisioning and policing is intricately tied to the edge-to-edge

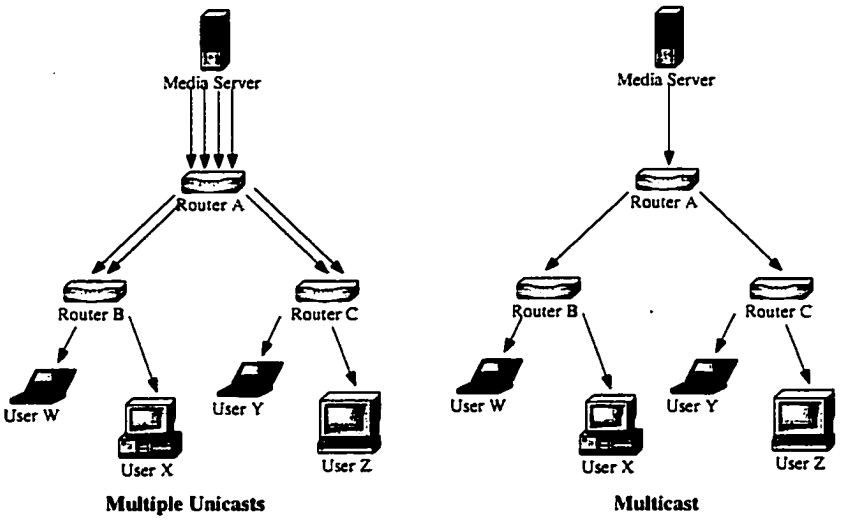


Figure 1.4 Group communication model

QoS experienced by flows across the network. Various works have focused on edge-wise traffic engineering [46], dynamic signaling [30], as well as other areas [12].

1.5 Multicasting

Although multicast was not specifically designed with the traditional notion of QoS in mind, the use of multicast can have a significant effect on QoS. Most notably, multicast can have a significant impact on the bandwidth utilization of many of the newly emerging applications that demand such QoS guarantees. For instance, in applications such as video-conferencing, on-line gaming, video/audio on-demand, or distributed workspaces, the group-oriented nature of such applications requires that the data be replicated to all members. Under the traditional unicast model, a group of N users requires that each piece of data be sent out N times from the source. This overhead is further compounded by the fact that such applications typically consume large amounts in the bandwidth due to their inherent rich multimedia nature.

Thus, the multicasting model was proposed for supporting such applications in an efficient manner [47]. Rather than setting up separate unicast connections for each member (receiver) in the group, a single group address is used to identify the entire group. Members that

wish to receive data from the group *join* the group and *leave* the group once they no longer wish to receive data. The sender or senders to the group transmit data to the multicast group address and rely on the network to disseminate the information to the members of the group. In essence, the multiple unicast connections are reduced into a single multicast tree that transmits/replicates information only when necessary. Thus for N users of the multicast service, significantly less than the bandwidth of N separate unicasts is required to deliver the packets to the end users (see Figure 1.4) [48, 49].

1.5.1 Characterizing Multicast

The difference between multicasting and separately unicasting data to several destinations is best captured by the *host group* model [50]:

"A host group is a set of network entities sharing a common identifying multicast address, all receiving any data packets addressed to this multicast address by senders (sources) that may or may not be members of the same group and have no knowledge of the groups' membership."

This definition implies that, from the sender's point of view, this model reduces the multicast service interface to a unicast one. The definition also allows the behavior of the group to be unrestricted in multiple dimensions: groups may have local (LAN) or global (WAN) membership, be transient or persistent in time, and have constant or varying membership. These definitions are expanded upon below:

- *Dense groups* have members on most of the links or subnets in the network whereas *sparse groups* have members only on a small number of widely separated links.
- *Open groups* are those in which the sender/source need not be a member of the group, whereas *closed groups* allow only members to send to the group.
- *Permanent groups* are those groups which exist forever or for a longer duration compared to the duration of *transient groups*.
- *Static groups* are those groups whose membership remains constant in time, whereas *dynamic groups* allow members to join/leave the group.

Although multicasting was designed to reduce the bandwidth utilization of group-oriented applications, multicast also increases the QoS to other users in the network by freeing additional bandwidth for use by other applications. If the network is not dramatically over-provisioned (i.e. in an extremely lightly loaded state), multicasting can offer a significant QoS improvement. As a result of this tremendous potential for bandwidth savings, a significant body of research has been devoted to multicasting [47, 51, 52, 53, 54].

1.5.2 State of Multicast

Despite the extensive body of research on multicast, the proliferation of applications taking advantage of multicast in the Internet has yet to emerge. While multicast support across the Internet has become fairly prolific across the backbone beyond the initial MBone implementation [47], multicast for the average user is still not available without considerable difficulty [55].

Several works have examined the relative proliferation of multicasting at various points during the history of multicasting [55, 56, 57]. The findings gathered during the summer of 2001 [55] present a fairly tepid adoption of multicasting across the vBNS (very-high speed Backbone Network Service) [58] which has been offering multicast service since 1995. The major findings of the study were as follows:

- *Tepid adoption:* At one point during the study, the number of active multicast state entries (for unique multicast groups) at the most active point (Chicago) was only 289. This represents a minimal change from the 199 entries recorded a full 2 years earlier that recorded the trends of adoption for a full 4.5 years in a study [57].

- *Prevalence of single sourced groups:* The majority of the traffic observed came from single-sourced multicast groups and only 1% of the groups observed had more than one simultaneous source. As a result, the study recommended removing the complexity incurred by multiply sourced multicast groups and adopting SSM (Source-Specific Multicasting) [59, 60].

- *Scalability of multicast routers:* When multicast begins to grow in popularity, there will be a tremendous growth in the underlying state information. If large numbers of groups are employed, there is the potential for destabilizing processing and memory requirements. Such destabilization has occurred in the past and given the relatively low utilization of multicast currently, an explosion in the use of multicasting before such support is deployed could incur serious consequences.

The findings of the study only accentuate the need for the solutions proposed by this dissertation, namely addressing the fundamental issue of the scalability of multicasting. A natural extension is to develop such solutions on the basis of the next-generation Internet, namely DiffServ. However, the integration of these two technologies present other unique challenges which provides the motivation for this dissertation. These challenges are discussed in more detail in the next chapter.

CHAPTER 2. BACKGROUND WORK AND THE PROBLEM

In this chapter, the primary conflicts between DiffServ and multicasting are described. The overviews of existing solutions as well as the proposed solutions are categorized and examined. Finally, additional areas for consideration when providing DiffServ multicasting functionality are discussed.

The chapter is organized as follows. First, Section 2.1 defines the conflicts between DiffServ and multicasting. Next, Section 2.2 introduces the three general classes of solutions to the DiffServ multicasting problem. Then, Section 2.3 presents existing solutions for the DiffServ multicasting problem and discusses the strengths and weaknesses of each approach. Following that, Section 2.4 describes a brief overview of the solutions proposed by this dissertation and a brief synopsis of their improvements versus existing solution. Finally, Section 2.5 concludes the chapter by discussing additional areas that the proposed solutions in the dissertation will address.

2.1 The DiffServ Multicasting Problem

The primary conflict between DiffServ and multicasting can be attributed to how DiffServ attempts to achieve scalability. Whereas DiffServ relies on only edge routers possessing intelligence and state information, multicasting relies on per-group state information throughout the entire network. Thus, when trying to integrate the two technologies, one is faced with two conflicting principles, core statelessness for scalability versus per-group state information for efficiency. The natural question is, which principle is more important, state information (storage and router complexity) or maximal network efficiency (bandwidth)?

It is the premise of this dissertation that state information is vastly more costly than

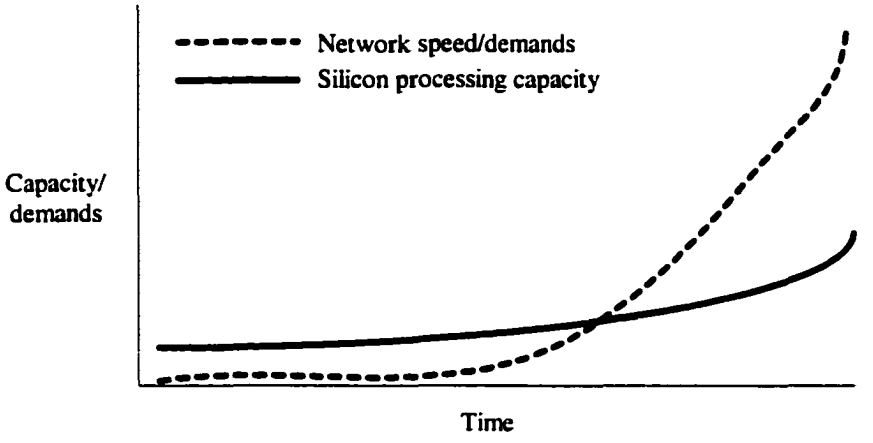


Figure 2.1 Growth rates of bandwidth versus CPU

bandwidth. Whereas one could argue that if one can increase bandwidth capacity, one can also increase the state storage capacity, such an argument is fundamentally flawed. First, the growths of bandwidth capacity and demand is far outpacing the improvements in CPU performance (see Figure 2.1 [38]). Hence, per-packet processing must be kept to an absolute minimum which is extremely difficult with per-group state (akin to per-flow state). Second, the increase in bandwidth only compounds the fact that it is the maintenance of state information (router complexity) and not necessarily storage that is the problem. The increase in bandwidth only exacerbates the problem in that increased bandwidth implies additional multicast packets and hence, the amount of per-group state information/group dynamics would increase as well. Thus, therein lies the multicast scalability problem whereby one cannot trade network efficiency at the cost of maintaining potentially unbounded per-group state information.

Although the problem of scalability in multicasting is not necessarily unique to DiffServ, the scalability problems of multicasting run counter to the fundamental principle of scalability that DiffServ was designed to address in the first place. The primary conflicts between DiffServ and multicasting are summarized below:

- *Scalability of per-group state:* The per-group state information of multicasting is not scalable to the high speeds of the core of the network. Thus, any solutions for DiffServ

multicasting must address this fundamental issue.

- *Sender versus receiver-driven QoS:* Whereas DiffServ provides a sender-driven QoS (packets are marked at the sender or ingress), multicasting is a receiver-driven service. As a result, appropriate mechanisms must be developed that bridge the gap between the sender-driven QoS of DiffServ and the receiver-driven QoS of multicasting.

- *Resource management:* Unlike unicast connections, a multicast packet may replicate into one or more packets in the network. This problem is compounded by the fact that the ingress node (entrance to the DiffServ domain) may not necessarily know the exact makeup of the multicast tree. Thus, appropriate signaling or management mechanisms must be introduced in order to manage the resource impact of multicasting on the network.

2.1.1 Scalability of Per-Group State

First and foremost, the primary conflict between DiffServ and multicasting lies in the per-group state information required for multicasting. For each multicast group flowing across the domain, each router must maintain as well as manage the state information (timers, group status, etc.). The maintenance of the group information is further compounded by the potential for a huge number of unique multicast groups.

Consider an example where an individual web server employs SSM (Source Specific Multicasting [60]). With SSM, the group identifier is expanded from the traditional IP multicast (no specific source) definition of only the group address $(*, G)$ to the tuple of the source address and the multicast group address (S, G). As a result, the multicast group address space expands significantly while the issues of address space contention and other complexities (shared trees, etc.) are removed.

Next, consider if the web server with the wealth of available group addresses is set up to place each web object as an individual group. Although this may seem far-fetched at first, such a scheme is not difficult to envision. Consider if one hundred, one thousand, or all of the web servers on the Internet employed such a scheme. If the traditional multicasting approach

was employed, the routers on the core of the Internet could face supporting and maintaining state information for millions or even billions of multicast groups. With current or even future processing capabilities, the cost of pushing such maintenance and support onto all of the routers on the core of the Internet would be prohibitive or even impossible. Thus, a simple question can be posed: Why not follow the DiffServ approach and minimize the number of routers that must maintain such state information? If bandwidth can be thought of as significantly cheaper than maintaining state information at all routers, can one leverage the DiffServ model to provide a better and more scalable multicast? These questions provide the motivation the work in this dissertation and are addressed in chapters 3 and 4.

2.1.2 Sender versus Receiver-Driven QoS

The second conflict between DiffServ and multicasting lies in the nature of how the QoS is selected. In DiffServ, QoS is provided through a sender-driven QoS. As packets are sent from the sender, the sender and ingress router for the domain mark/shape the packet. In contrast, multicasting is built upon a receiver-driven QoS. As only the receiver knows its own QoS requirements and thus the receiver is the only entity that can drive the appropriate QoS. Hence, there exists a conflict in how QoS is delivered[1].

If QoS is pushed entirely from the sender, one is faced with the dilemma of a one-size-fits-all scenario whereby all group members must either subscribe to the QoS level or not subscribe at all. This scheme is the simplest but comes at the cost of flexibility. If the current capabilities of users on the Internet are any indication of future trends, the cost of inflexibility may be the alienation of many potential users of the application or the service itself. The other extreme is to use separate groups for each level of QoS requested by the receivers. Such a scheme meets the QoS levels for the receivers but is burdened by the bandwidth costs of each additional tree for the additional QoS levels.

An alternative scheme is to satisfy the QoS requirements of heterogeneous multicast receivers without requiring separate trees for each QoS level. In such an approach, the QoS

[1]Note that this is only the QoS regarding network specifics (loss, delay, fault tolerance, jitter), not the content of the data (layered multicasting [61], heterogeneous data rates (56k, DSL, T1), etc.).

level changes as packets pass across the multicast tree. The unique aspects of this scheme is investigated in more detail in Chapter 5.

2.1.3 Resource Management

The third conflict between DiffServ and multicasting lies in the aspects of resource management. Although such aspects could be considered to be conflicts with standard IP multicasting, the requirements for policing and resource management in DiffServ introduce several unique problems. In short, the conflict regarding resource management can be reduced to the problem of packet replication [39]. Under unicast, one packet in yields one packet out. However with multicasting, this is may not be the case as one packet entering the domain may become multiple packets out. Hence, the actual resource consumption may vary somewhere between the cost of separate unicasts across the domain and the cost of a single unicast (no branching in the multicast tree) [48, 49]. In order to efficiently allocate network resources, one needs to be able to quantify the resource impact of a multicast group on the DiffServ domain and appropriately shape/police multicast packets at the ingress routers.

Without source-wise policing of the group, traffic policing is an extremely difficult if not impossible task. The problem simply stated is, how does one police shared bandwidth that can be consumed at any of N ingress routers for the group? In fact, multicast provisioning today is considered something akin to a medieval "black art" where relatively little about the problem is understood and is a topic for future research. Although several architectures have studied the allocation of such resources for single sourced groups [42, 62, 63], the introduction of an architecture for both provisioning and policing of multiply sourced groups has yet to appear.

A secondary conflict arises with how information for member join/leave is conveyed for resource allocation. In [40], it was noted that a resource reservation problem called the NRS (Neglected Reservation Subtree) problem can occur if member join (graft) messages are not approved for resource allocation before data is transmitted on the branch (see Figure 2.2). The NRS problem finds its basis in the distributed nature of the multicast tree and the separation of multicast routing/replication from the ingress-based policing/allocation of DiffServ. Since

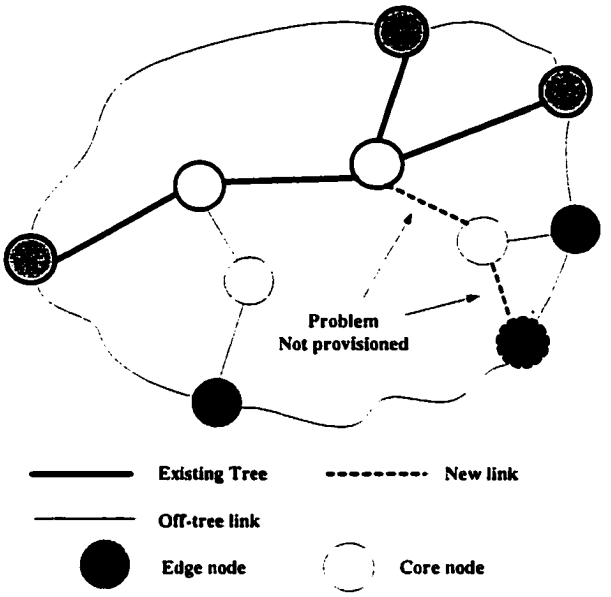


Figure 2.2 Neglected Reservation Subtree (NRS) problem

an edge node does not necessarily know the makeup of the multicast trees that may exist in the domain, it cannot determine the resource requirements to graft onto the multicast tree. As a result, each router along the path (both the edge routers and the core routers) for a new join request must appropriately reserve resources for the new branch. The requirement for core routers to negotiate for resources represents a fundamental shift from simple, stateless core routers. Although the primary goal of the proposed solutions in chapters 3 and 4 is scalability, the solutions also offer significant benefits in terms of resource management. The discussion in each chapter includes commentary regarding resource management of the respective architectures.

2.2 Overview of Approaches

In order to offer support for DiffServ multicasting, the solutions can be divided into three main classes, state-based, edge-based, and encapsulation-based. The approaches are summarized in Table 2.1 and discussed in more detail below.

Table 2.1 Summary of DiffServ multicast approaches

Type	**Replication Allowed**	**Replication Information**	**Strengths (+)** *Weaknesses (-)*
State-based	Everywhere	Per-group state in routing table	+ Most efficient B/W usage - *Scalability* - *State in DS core*
Edge-based	Edge routers	None Tunneling [64]	+ Simple + Meets DS requirements - *Dense groups*
Encapsulation-based	Everywhere	Included in packet	+ Scalable + Meets DS requirements - *Processing cost* - *Per-packet overhead*

2.2.1 State-Based Approach (Traditional IP Multicast)

In this approach, the per-group state information for each of the multicast groups is maintained in the edge routers as well as the core routers of the DS domain. When a new egress point (router where the packet exits the domain) wishes to join or leave a multicast group, the corresponding core and edge routers propagate upstream their state information for the multicast group that is affected. For a join request, resource reservation is handled at each router (core or edge) until the request reaches the multicast tree. This approach is the equivalent of simply applying the traditional IP multicast model without distinction to the edge or core routers of the DiffServ domain.

To start, this approach contradicts the fundamental concept of stateless core routers in the DiffServ architecture. This approach has scalability problems because of the complexity of per-group state maintenance at all core routers. In addition, the state-based approach also pushes the complexity of multicasting protocols onto the core (control messages, resource reservation), thus violating the concept of simple core routers. The state-based approach is viable only if the number of unique multicast group stays extremely small such as currently seen today.

2.2.2 Edge-Based Approach

The second approach is to restrict the location of multicast capable routers within the DiffServ domain. Rather than allowing all routers to be multicast capable, only the edge routers of a DiffServ domain are allowed to be multicast capable. This approach is simple to implement and is highly scalable but incurs performance degradation in terms of bandwidth consumption versus the state-based approach. Apart from its simplicity, it is highly suitable for sparse groups where replication of packets at the core routers is less likely to occur compared to that of the dense groups.

In such a solution, core routers are entirely multicast unaware, thus requiring zero support for multicasting (replication or control messages). Thus, there is no additional implementation necessary in the core routers and the per-group statelessness of the edge-based approach complies with the goals of the DiffServ architecture.

2.2.3 Encapsulation-Based Approach

The third approach is to embed the multicast information within the packet itself. In the encapsulation-based approach, the multicast tree for the domain is encapsulated within the multicast packet. Similar to how core routers rely on the marking of the packet to determine the PHB for unicast QoS, the encapsulation-based approach adopts a similar methodology whereby the necessary multicast information is embedded inside the packet. Hence, this approach does not require per-group state information and fits extremely well within the DiffServ architecture.

However, encapsulation does introduce several additional costs/overheads. First, encapsulation consumes additional bandwidth for each packet in the form of encoding the multicast tree and thus may not scale well with the group size. This scalability problem is alleviated by encoding only the tree for the domain, not the entire end-to-end multicast tree. As a result, the size of the tree is dependent upon only the number of egress points for the domain, not the number of actual receivers in the multicast group. The second tradeoff that occurs with encapsulation is the additional CPU cost incurred due to header processing that may be entirely alleviated through hardware support.

2.3 Existing Solutions and Related Work

To date, there has been a relatively small amount of research devoted to DiffServ multicasting. Several of the recent works are summarized below and discussed in more detail in the subsequent chapters as applicable. Except for the last work (QDM) which is a hybrid between edge-based approach and the state-based approach, each of the following works falls into the first category, the state-based approach.

2.3.1 IETF DiffServ Workgroup Drafts

In [40], the authors investigated the issues of supporting multicast using traditional IP multicast in a DiffServ environment. Specifically, the authors outlined two key problems, proper allocation of resources (NRS problem) and support for heterogeneous QoS. The support for heterogeneous QoS was first outlined in [65] before being incorporated into [40]. The initial research paper for the work appeared in [66]. In addition, many of the final decisions regarding DiffServ multicasting were deferred to other IETF working groups such as the NSIS (Next Step in Internet Signaling) working group.

However, these works suffered from several key weaknesses. First, the work rests upon the assumption that traditional IP multicast is scalable. In fact, such an argument is mentioned explicitly in [40] as a reason for not addressing the scalability issue. This assumption is valid only for the current limited usage of multicasting, not for multicasting to become a ubitiquous network service. Second, the proposed solution for heterogeneous QoS is dramatically oversimplified and is discussed in more detail in Chapter 5.

2.3.2 QUASIMODO: QUAlity of ServIce-aware Multicast Over DiffServ and Overlay Networks

In [67], the authors investigated the practical issues regarding support for PIM (Protocol Independent Multicast) in the Sparse Mode (SM) [68, 69] in a DiffServ environment. The authors proposed extensions to the PIM-SM state model along with the use of the GRIP (Gauge & Gate Reservation with Independent Probing [70]) model to offer dynamic resource alloca-

tion. Most notably, the solution introduced a new QoS state (QoS/No QoS) for supporting heterogeneous QoS in the network.

Similar to the previous work, this work also suffers from the same problem of ignoring the fundamental conflict of scalability. Although the work explores the issue of heterogeneous QoS more than the previous work, the discussions regarding heterogeneous QoS were not discussed in more detail due to space restrictions.

2.3.3 DAM: DiffServ Aware Multicasting

In [41], the authors proposed a scheme for addressing the two issues of resource management (the NRS problem) and receiver-driven QoS. The DAM model incorporates three key concepts: weighted traffic conditioning, receiver-initiated marking, and heterogeneous DSCP encapsulation. The main contribution of the work is the weighted traffic conditioner as the other two concepts offer minimal improvement over the initial IETF work.

2.3.4 QDM: QoS-aware Multicasting in DiffServ domains

The work in [71] addresses the issue of QoS-aware multicast routing in a DiffServ environment through a model called QDM. In the QDM model, routing decisions are migrated entirely to the edge of the network, thus decoupling the core from control messages and resource negotiations. The routing decisions are made using a modified version of Dijkstra's routing algorithm and packets are transmitted using REUNITE [72], a recursive unicast approach for multicasting. Although this work addresses one of the fundamental issues of DiffServ multicasting, the removal of control complexity from the core, it does not completely remove multicast state from the core. In addition, the work does not address the routing issues associated with heterogeneous QoS.

2.4 Proposed Solutions

Although the existing solutions address different aspects of the conflicts between DiffServ and multicasting, none of the existing solutions address all three conflicts. In the next two

chapters, this dissertation proposes two architectures that address all three of the root conflicts between DiffServ and multicasting. Chapters 5 and 6 expand upon the two architectures by addressing issues such as heterogeneous QoS and fault tolerance. The next four chapters are briefly summarized below in relation to the material presented in this chapter.

2.4.1 DSMCast

The first architecture, DSMCast (DiffServ MultiCast) [73, 74], proposes an encapsulation-based solution that moves the burden of state information from the router to the actual multicast packet itself. The tree information is added as an extension header at the ingress (entrance) to the DiffServ domain. The tree header allows the ingress router to explicitly specify the router of the packet and consequentially know the exact resource consumption of the packet for policing. A heterogeneous QoS extension allows DSMCast to use a single tree while still meeting the heterogeneous QoS needs of the egress points in the multicast tree. The primary strength of this architecture is that it addresses all three of the conflicts with performance close to that of the state-based approach. However, the key weakness of the approach is that it does introduce additional overhead as well as requiring additional hardware for processing the DSMCast extension header.

2.4.2 EBM

The next architecture, EBM (Edge-Based Multicasting) [75], proposes a solution that relies on only edge routers for replication of packets in the network. Packets are tunneled between edge nodes, thus entirely removing the concept of multicasting from the core of the network with only a marginal performance penalty. The distribution tree is constructed using an algorithm, called the Edge Cluster Tree, that uniquely captures the qualities of heterogeneous QoS routing. Similar to DSMCast, EBM also addresses all three of the conflicts between DiffServ and multicasting. The EBM architecture trades the penalty of additional bandwidth consumption in order to remove the hardware requirement of DSMCast.

2.4.3 Extensions for Heterogeneous QoS

Next, the dissertation focuses on how heterogeneous QoS impacts the DiffServ environment [76]. The chapter presents the first in-depth study of the use of variable QoS levels (dynamic PHBs) in a single multicast tree. The chapter investigates the implications of heterogeneous QoS in traditional IP multicast, DSMCast, as well as several of the existing solutions. Most notably, the chapter investigates the concept of the *Good Neighbor Effect*, a unique effect that occurs from the use of a single multicast tree and its impact on both users as well as service providers.

2.4.4 EI-HELLO

The fourth major component of the dissertation examines the effect of link and router faults on the proposed architectures. The chapter proposes a solution called Edge-based Intelligent HELLOs (EI-HELLO) to exploit the unique aspects of the DiffServ architecture to help expedite the fault detection of link state protocols [77]. The EI-HELLO model represents a novel application of DSMCast that has benefits for fault detection and network management.

2.5 Improving Multicasting

In addition to addressing the fundamental conflicts between DiffServ and multicasting, the secondary goal of this dissertation is to offer benefits for the general adoption of multicast. Several of the benefits of the proposed solutions include:

- *Simplified deployment:* Rather than focusing on an end-to-end solution to the multicast problem, the proposed architectures reduce the problem to a more tractable edge-to-edge domain transport problem.

- *Security:* By coupling multicasting with policing and resource management, the threat of multicast as a Denial of Service (DoS) attack can be reduced. In addition, security can be included on a local level rather than relying on only end-to-end support.

- *Performance enhancements:* The removal of state information from the network can offer a significant performance increase for core routers. The proposed solutions will not only meet the scalability demands of the future but also can offer significant speed improvements in the near-term.

2.5.1 Simplified Deployment

As discussed earlier, the adoption of multicast has been fairly tepid [55]. Multicast deployment is faced with the classic *chicken versus egg* dilemma between ISPs and application developers. To start, ISPs will not fully deploy multicast until customers demand such services. However, customers will not demand such services until applications take advantage of them. Application developers will not write applications demanding multicast until such network services are ubitiquous. Thus, the adoption of multicast has stalled beyond several early adopters. Unfortunately, without an overwhelming incentive beyond the current marginal solutions, very few will undertake the tedious process of rolling out Internet-wide support for multicasting.

One of the major benefits of the proposed solutions is to reduce the difficulty of deployment. Rather than worrying about multicast functionality throughout the network, the proposed solutions transform the multicast problem into a transport problem, a problem ISPs are quite familiar with. An ISP need only worry about basic multicast transport, not the intricacies of supporting each protocol through its entire network.

2.5.2 Security

One of the security fears regarding multicasting is that multicasting can become the ultimate Denial of Service (DoS) attack. Without proper policing, an attacker can flood a multicast group with traffic, thus launching a huge DoS attack with the unwitting assistance of the network. Although there have been very few documented issues with multicast security, the security of multicasting remains relatively untested due to the limited propagation of multicast to the majority of the end-users of the Internet. Despite the fact that multicast is an appealing target for DoS attacks, the limited propagation and adoption of multicasting has

kept it from being a large security problem as of yet.

In the two proposed architectures, security may be added on a domain-wise basis rather than requiring end-to-end cooperation. The key benefit is that such security can be added on a domain level (individual ISP) and operate entirely transparent to those outside the domain. An ISP can force authentication and encryption onto all member join/leave requests to prevent common routing attacks. Since the two architectures change multicasting to a transport service, multicast packets using the transport protocol can only originate from edge routers in the domain, thus offering another opportunity to deny external attacks. In addition, the inherent shaping/policing nature of the edge routers reduces the potential of successful attacks on a successfully configured network.

2.5.3 Performance Enhancements

A final obstacle to multicast deployment is the performance decrease that occurs when multicasting is enabled in routers in the network. In several commercially available routers, one can observe performance levels decreasing on the order of 10-15%. The performance decrease can be attributed to the separate inspection/processing that must occur for multicast packets that adversely affects all packets passing through the router. Thus, although an ISP may gain bandwidth from the savings of multicast, the additional cost of deploying not only multicast-capable routers (software support) but also faster routers (processor speed/throughput), may not be offset by the customer demand. As a result, despite the fact that multicast is supported by many of the routers currently available, the multicast functionality is widely disabled.

By addressing the fundamental issue of scalability, the two proposed architectures also address the issue of performance. Without per-group state overhead, core routers will be able to operate without any impact from the number of unique multicast groups present in the network nor be affected by the complexity associated with multicast protocols. Due to the fact that the core routers will naturally be expected to run at higher speeds (by virtue of being the backbone of the network), performance improvements in the core can offer significant benefits.

CHAPTER 3. DSMCAST: STATELESSNESS THROUGH ENCAPSULATION

In this chapter, an encapsulation-based approach for DiffServ multicasting, DSMCast (DiffServ MultiCast), is proposed. The primary goal of DSMCast is to address the issue of per-group state information in the core. By migrating the per-group state information from the router to the packet, core routers are allowed to remove the complexity of multicast routing and multicast state maintenance. The migration of state information to the edge has implications for not only core scalability but also for resource management and heterogeneous QoS. Each of these three issues is discussed in the chapter along with the basic mechanics of group dynamics (member join/leave).

The rest of the chapter is organized as follows. The first section, Section 3.1, describes the fundamental case for an encapsulation-based approach versus the existing solutions. Next, Section 3.2 outlines the basic concepts of the DSMCast architecture regarding how the multicast transport service is delivered (packet header structure and tree construction). Section 3.3 details the join/leave protocol in both SSM and traditional IP multicast environments. Following that, Section 3.4 examines DSMCast as it relates to other protocols and existing models. Then, Section 3.5 analyzes the performance of DSMCast through both theoretical and simulated results. Finally, Section 3.7 concludes the chapter by offering several concluding remarks regarding DSMCast.

3.1 A Case for an Encapsulation-Based Approach

The basis for the architecture presented in this chapter can be found in the underlying encapsulation-based approach for multicast transport across the DiffServ (DS) domain. Unlike

the state-based approach of traditional IP multicasting, the encapsulation-based approach can potentially satisfy the three conflicts between DiffServ and multicasting. Thus, although an encapsulation-based approach does incur several performance penalties, it is the premise of this dissertation that the benefits of such an approach far outweigh the penalties. In fact, the benefits of the encapsulation-based approach go well beyond the three primary conflicts of DiffServ multicasting and such benefits are summarized below:

- *No per-group state:* An encapsulation-based approach does not require per-group state information within core routers, thus meeting the DiffServ goal of stateless core routers.
- *Hardware support:* The CPU cost for supporting an encapsulation-based approach may be reduced to zero through the addition of hardware support.
- *Protocol independence:* An encapsulation-based approach shields core nodes from the complexities of the outlying multicast routing protocols reducing the multicast problem in the core to a transport problem.
- *Resource management:* The edge router is able to easily monitor the resources being consumed by the multicast tree since the router is responsible for determining the exact path the multicast tree takes in the core of the DS domain.
- *Heterogeneous QoS support:* Rather than requiring separate trees for each level of QoS, encapsulation can allow the PHB (Per-Hop Behavior) to change as the multicast packet crosses the domain to support heterogeneous QoS in a single multicast tree.
- *Deployment:* Rather than requiring end-to-end support, the transport nature of the encapsulation-based approach reduces the scope of the deployment problem to a domain-wise problem whose operation is transparent to the external Internet.

However, there are also several weaknesses to the approach which must also be considered as well.

- *Overhead:* The savings from the embedded header may not offset the transmission cost of the header versus separately unicasting/tunneling packets over the DS domain.

- *Edge Scalability:* Edge routers must maintain per-group state information. Although these routers are better suited for such information, this is a non-trivial problem which is inherent to multicasting.

- *Network Faults:* The routers in the network may fail or the network may be reconfigured, thus possibly requiring the edge routers to detect such changes and adapt the multicast tree to changes in the network topology.

3.2 DSMCast - An Approach for DiffServ Multicast

The primary goal of the DSMCast architecture is to address the three primary conflicts between DiffServ and multicasting, namely core statelessness, heterogeneous QoS, and resource management. The encapsulation-based approach represents an ideal approach for satisfying these three conflicts with a minimal amount of overhead. Although the encapsulation-based approach can address all three issues, the use of encapsulation and removal of core state introduce several challenges that must be addressed.

To start, the first challenge is to develop an efficient packet header that conveys the multicast tree and is optimized for fast hardware processing. Next, the header must support options to accommodate heterogeneous egress points with minimal bandwidth overhead. Furthermore, the construction and maintenance of the tree and the packet header must also be addressed. Finally, a new domain-wise join/leave protocol must be designed to overcome the statelessness of core routers. Additionally, although the general trend in multicast is towards SSM [60], a complete architecture should also be able to offer support for traditional IP multicast as well as SSM.

Hence, it is the motivation of this chapter to address these challenges in the DSMCast architecture. In short, the DSMCast architecture can be subdivided into four main components, the transport mechanism, the extensions for heterogeneous QoS, the tree construction mechanism, and the egress (member) join/leave protocol.

3.2.1 Transport Mechanism

The most basic function of the DSMCast architecture is to provide a core stateless transport mechanism for multicast traffic across the domain. To accomplish this goal, all replication/routing for multicast packets is controlled by the DSMCast *Tree Encapsulation Header* (TEH). When a multicast packet arrives at the ingress router (entrance) to the DS domain, the ingress router must insert the TEH into the multicast packet. The ingress router looks up the TEH for the specific multicast group (S, G) and inserts the TEH between the layer 3 (IP) and layer 4 (UDP, etc.) headers of the packet. This location is chosen as it preserves the standard IP routing and DiffServ classification. In addition, since IP options are a rare occurrence, the DSMCast TEH can be thought of as being located in a fixed location[1].

In fact, unless the core routers in the domain rely on the layer 3 information for packet length or do layer 4 shaping/routing (rather than the inter-packet gap), the TEH may be inserted without modifying any of the remaining fields of the packet. Since the TEH is only added at the ingress of the domain and removed at the egress, the TEH simply appears as packet data to the IP routing mechanisms of core routers.

Upon receiving a multicast packet, the core router will give the packet to the hardware mechanism for DSMCast[2]. The hardware mechanism will inspect the packet to determine which interfaces the packet should be replicated upon. Once replicated, the packet will be placed in the standard DiffServ unicast class queues and treated according to the PHB specified by the DSCP (DiffServ CodePoint). If necessary, the DSCP of the packet may be modified to support heterogeneous QoS as specified in the TEH. Once given to the unicast queue, the multicast packet is treated identically to all other unicast packets at the router.

The only tasks that must be performed by the hardware mechanism are the location of the tree information for the specific router, the marking of the DSCP of the packet[3], and the replication the packet. These simple operations of pattern matching (information location),

[1] In the case where IP options are present, many routers divert the packet to a slow path (software) for processing.

[2] Alternatively, such support may be offered in software at the cost of significant performance.

[3] Additionally, the IP checksum of the packet may need to be updated as well which is similar to the modifications of the IP checksum field for the TTL.

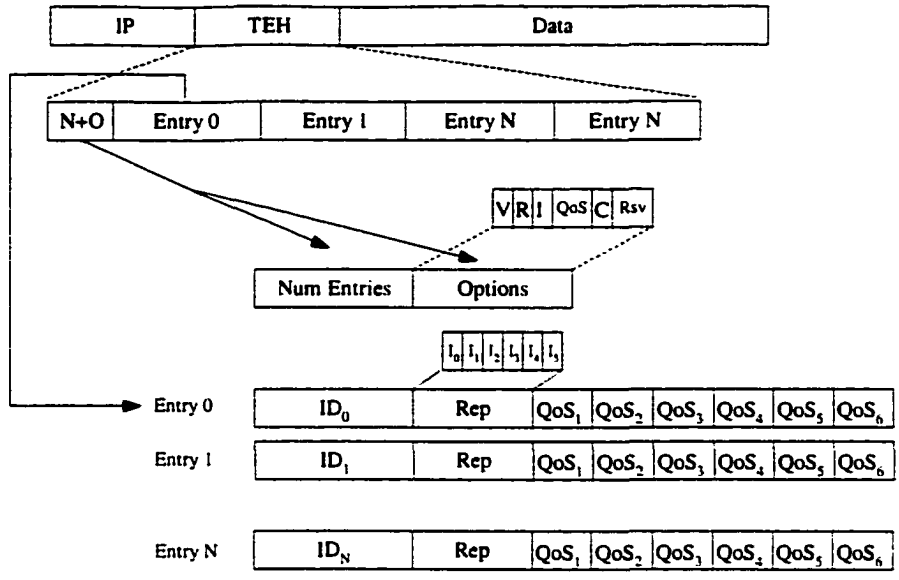


Figure 3.1 DSMCast Tree Encapsulation Header (TEH)

bit marking (DSCP, checksum), and packet replication are easily accomplished by the vast majority of commercially available network processors at current and future network speeds [78].

3.2.2 Tree Encapsulation Header (TEH)

The Tree Encapsulation Header is the sequence of bits that are responsible for routing and replicating the packet across the core of the DiffServ domain. The makeup of these bits is faced with two potentially conflicting goals: optimal bandwidth usage versus processing speed. If a software-based network processor is used, the search time for the header can become a performance bottleneck. However, an alternative approach is to use a CAM (Content Addressable Memory [79]) whereby many patterns are compared at once. An FPGA could also be used to accomplish the same task.

The TEH consists of three fields, the *NumEntries* field (number of entries in the header), the *Options* field, and the replication/routing entries. The fields are a fixed length in order to allow for easier processing by the hardware mechanism. Figure 3.1 shows the layout of the

Table 3.1 Tree Encapsulation Header Options field

Bit(s)	Abbreviation	Field	Description
7	V	Variable QoS	0 - No variable QoS 1 - Variable QoS present (heterogeneous QoS)
6	R	Extended Replication (Rep/QoS) bits	0 - 5/2 bits per entry (24 bits) 1 - 6/3 bits per entry (32 bits)
5	I	Extended ID	0 - Use an 8 bit ID (default) 1 - Use a 16 bit ID
4,3	QoS	QoS Setting	Switch for QoS transformation methods
2	C	Clear Rep	0 - Leave replication bit present 1 - Clear replication bits on replication
1,0	Rsv	Reserved	Unused

DSMCast TEH.

The first field, the *NumEntries* field, is an 8 bit field that contains the number of entries present in the TEH itself. For each non-leaf node of the multicast tree, an entry will be present. Next, the *Options* field is used to convey the configuration of the TEH. The settings for the options field are listed in Table 3.1. The next portion of the TEH is the replication/routing information. Depending upon the setting in the options field, each entry may be either 24 bits (3 bytes) in the 5/2 case (5 interface replication bits, 2 QoS transformation bits per interface) or 32 bits (4 bytes) long in the 6/3 case. Each entry consists of a unique identifier (8 bits), the replication information (5 or 6 bits), and the QoS transformation code (if included).

By default, DSMCast assumes that the unique identifier will be the last 8 bits of the IP address of the router. Alternatively, the network administrator may assign a unique ID instead to each router. The re-use of the IP address allows for easy integration from the OSPF/IS-IS topology information and should be sufficiently unique for an individual router in the domain. In the event that a longer unique address is required, an extension bit can be enabled to double the unique ID space to 16 bits. If a unique ID separate from the IP address is used, a complete list of mapping between the IP address and unique ID must be given to all routers in the domain.

The replication field denotes the interfaces on which the packet should be replicated. The order of the interfaces is sorted by the unique ID of the neighbor on other side of the link corresponding with the interface. The maximum number of interfaces is denoted by the Extended

Table 3.2 Example QoS transformation table

Bits	Description	Bits	Description
000	No change	100	Change to Best Effort
001	-1 delay class	101	-2 delay classes
010	-1 loss class	110	-2 loss classes
011	-1 delay/-1 loss	111	-2 delay/-2 loss

Replication bit of the options field (5 or 6 interfaces).

During operation, a core router will locate its entry in the DSMCast header through the ID field. The *Rep* field will denote how the packet should be replicated (1=Replicate. 0=No Replication). If present, the QoS transformation code denotes how to change the DSCP (and hence the PHB). A complete example of the TEH is given in Section 3.2.4.

3.2.3 Variable QoS Extensions

In order to support heterogeneous QoS within the multicast tree, the DS field must be allowed to change as the multicast packet is replicated. In such a heterogeneous QoS tree, the "best" PHB is propagated up the tree towards the ingress router. As a result, the PHB can only "worsen" as the packet passes down the multicast tree[4]. The *Variable QoS* option adds a *QoS Transformation* field for each bit of the replication field. In the QoS transformation field, a code is given that determines how the current PHB should be changed. The actual transformation process can be configured as desired by the network administrator. Table 3.2 lists an example of an encoding using 3 bits. For example, a bit field of 111 would change an AF10 packet by reducing the DSCP by two loss and two delay classes to AF32. The intricacies associated with heterogeneous QoS management are discussed in greater detail in Chapter 5.

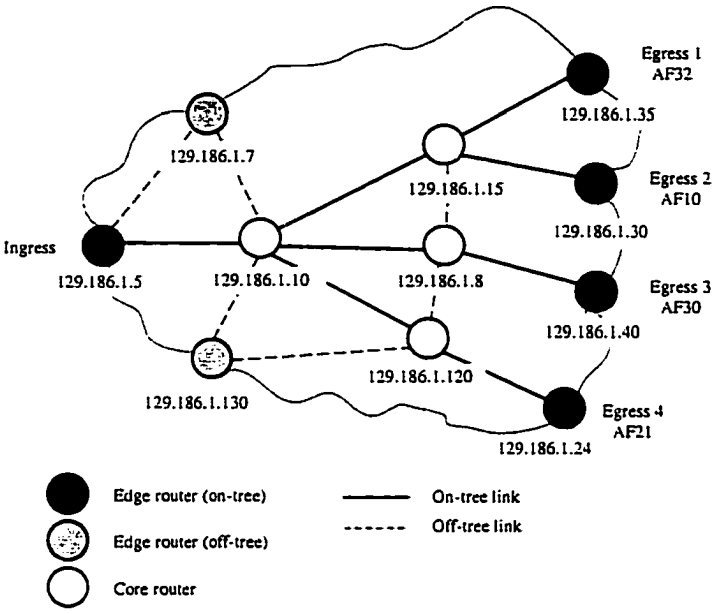


Figure 3.2 DSMCast example network

3.2.4 Example

Consider the DS domain and multicast tree listed in Figure 3.2. The ingress router for the group is at IP address 129.186.1.5. The group has a total of 4 egress points, each with different QoS requirements. All of the egress points are requesting the Assured Forwarding denoted by $AFxy$ where x is the delay class, y is the loss class, and a lower number denotes a higher priority class. The multicast tree for the domain and the PHBs of the respective links are listed in Figure 3.3.

For each router in the multicast tree that is not a leaf node, a replication/routing entry must be generated. Figure 3.4 shows the entry for the core router at 129.186.1.10. At this node, the *Rep* field has three ones present in order to replicate the packet onto the interfaces for the downstream nodes of 129.186.1.8, 129.186.1.15, and 129.186.1.120. For each of the possible interfaces, a QoS transformation code is also included. For the link to 129.186.1.15, an entry of 000 is used to denote no change to the PHB. In contrast, the entry to 129.186.1.8

[4]The notion of better/worse PHBs depends upon many factors (current network state, rate-limiting, etc.) and is best determined by the network administrator.

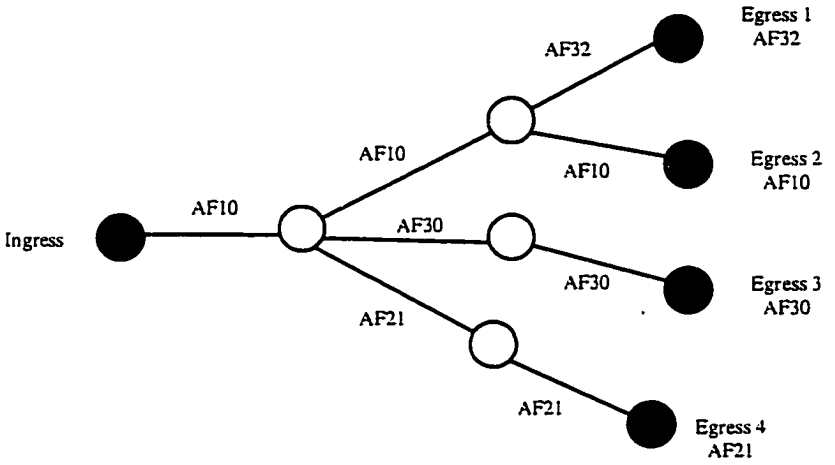


Figure 3.3 Breakdown of heterogeneous QoS support

uses an entry of 101 to denote a change to AF30. Note that the final egress points do not need to be included in the packet as the packet is at a leaf node and hence is no longer under the jurisdiction of DSMCast regarding replication and routing. The final DSMCast TEH is listed in Figure 3.5.

3.2.5 Tree Construction

The TEH is constructed and maintained by the ingress router of the multicast group. It is assumed that each edge router has complete topological knowledge of the entire DS domain (core and edge routers). This information is easily available from common intra-domain protocols such as OSPF [80] or IS-IS [81] or by configuration of the network administrator. Note that this information is only the topology of the local DS domain, not the topology of the global Internet nor the resource allocations on each of link of the domain.

As a result of its knowledge of the DS domain, the ingress router can appropriately construct a multicast distribution tree for a specific multicast group and update the tree as egress routers join and leave the multicast group. The tree size is strictly limited by the number of egress routers that wish to receiver traffic and is independent of the number of actual receivers on the global multicast tree. The tree construction algorithm is left to the decision of the network

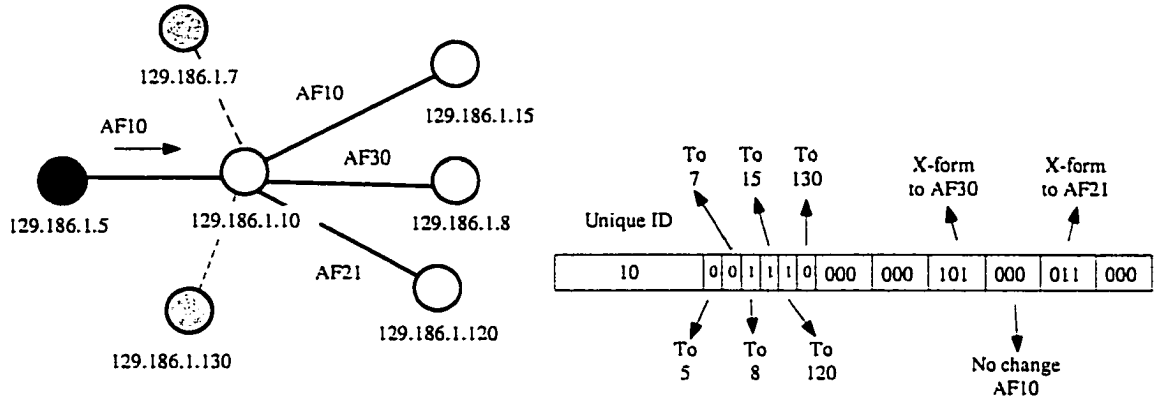


Figure 3.4 Example entry in the DSMCast Tree Encapsulation Header (TEH)

5	1	1	0	00	1	00						
5	0	1	0	0	0	0	000	000	000	000	000	000
8	0	0	1	0	0	0	000	000	000	000	000	000
10	0	0	1	1	1	0	000	000	101	000	011	000
15	0	0	1	1	0	0	000	000	000	111	000	000
120	0	0	1	0	0	0	000	000	000	000	000	000

Figure 3.5 Final DSMCast Tree Encapsulation Header (TEH)

administrator and may be optimized either for cost (shared tree [82]) or QoS (shortest path tree [83]). Although the routing of the tree may be based on network dynamics (due to the fact that the tree is route-pinned inside the packet), such methods are left as a topic for future research.

In addition, it is assumed that there is an underlying scheme for detecting faults or reconfigurations in the underlying domain topology. The issues with fault detection/recovery for a distributed set of edge nodes and stateless core nodes are beyond the scope of this chapter but are discussed in Chapter 6. While the route-pinned packet does potentially make DSMCast briefly more susceptible to faults, it does offer several significant benefits. First, the route-pinned nature of the tree allows the ingress router to know the maximum resource consumption by a given multicast tree. Since the ingress router must explicitly specify each link for replication and the PHB associated with each link, the resource consumption is available at the router that does the policing in the first place. Second, admission control for the tree is dramatically simplified. Since the ingress router knows the exact makeup of the tree, it can negotiate for the resources of the tree on a whole rather than leaving the tree to be negotiated piece-wise by individual routers along the tree. Whereas resource management in the state-based approach can be complex and difficult, resource management is significantly easier with DSMCast as a result of only the ingress router maintaining resources for the multicast tree.

In addition to fault tolerance, another potential cause for concern lies with the size of the TEH due to the number of routers in the domain. This argument can be countered in several respects. To start, the size of the header is directly related to the number of core routers in the multicast tree, not the number of egress points nor the number of receivers in the multicast group. Furthermore, the multicast tree is the distribution tree for transport across a single domain, not the global multicast tree. If necessary, the tree may be subdivided into multiple trees to reduce the size of the DSMCast TEH.

3.3 Egress Router Join/Leave

The next logical step after the providing the underlying transport service is to solve the problem of how join/leave messages are processed. Whereas the state-based approach simply relied on individual nodes propagating join/leave messages towards the multicast tree, the removal of state information and control packet processing from the core leaves a void that multicast control cannot cross without intervention.

The first router that will receive a join or leave message must be an edge router due to the DiffServ architecture. At that point, it is up to the edge router to determine where to forward the control packet. In the case of a leave packet, the task is fairly simple as the edge router is already part of the multicast tree for the group and hence knows the ingress router for the group. In contrast, the join process is somewhat more difficult depending upon the underlying multicast model. If the traditional multicast model is employed, the process can be significantly more difficult as the group address $(*, G)$ does not imply the location. As a result, the multicast group may be present anywhere in the global Internet, thus necessitating a search mechanism for the multicast group. Several schemes such as MSDP (Multicast Source Discovery Protocol [84]), MBGP (Multicast Border Gateway Protocol) [85, 86] have been developed to address this. The egress join problem (see Figure 3.6) can be summarized thusly:

For a join request to group G_x from downstream node R_y that arrives at edge node (E_{Rcvr}), determine the appropriate edge node (E_{Egress}) to which the multicast join request should be forwarded to and propagated outwards from the DS domain such that the QoS constraints of the join request are met.

However, if the location of the group is tied to an actual source IP address (S, G), the problem of location vagueness disappears and the problem simply reduces to basic IP routing. For multicast models such as SSM and CBT (Core-Based Trees) where the location is required, the complexity of multicast routing decreases dramatically. In fact, the additional complexities of traditional IP multicast offer little benefit that cannot be accomplished by SSM. A recent study on multicasting noted that very few applications take advantage of the multi-sourced

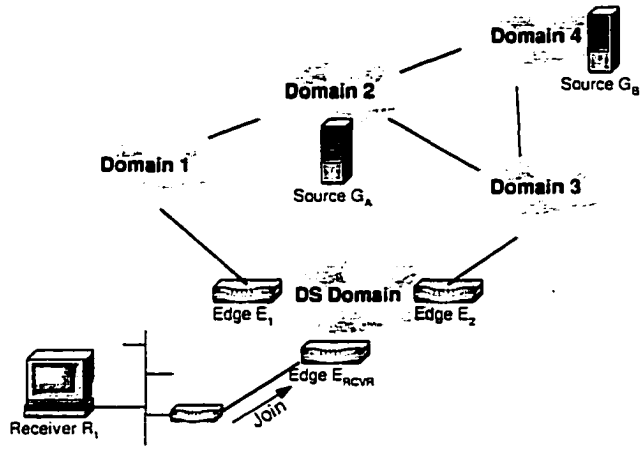


Figure 3.6 Multicast join routing problem

potential of traditional IP multicasting [55]. As a result, the join/leave protocol for DSMCast is optimized for an SSM environment. However, as legacy applications may still require non-specific source support, extensions for traditional IP multicasting to DSMCast are included as well.

3.3.1 Egress Join - SSM

When a join request arrives at an edge router, the request will include a source/group identifier (S, G). The edge router must then forward the packet to the appropriate ingress node for multicast source. Due to the fact that IP routing provides only for the next hop to a given destination IP, ascertaining the ingress point for an arbitrary IP may be difficult if not impossible. The solution to this problem is to simply tunnel [64] the join request towards the source of the group. As a result, the egress router can rely on standard IP routing rather than a special routing mechanism. In the tunnel header, a special identifier will be recognized by the ingress router. Upon receiving the packet, the ingress router would intercept and process the packet rather than forwarding the packet.

Similar to how DSMCast provides support for both SSM as well as traditional IP multicasting, DSMCast also provides flexibility in how the egress router joins the multicast group.

Rather than forcing a router to explicitly specify a PHB with the initial join, an egress router may first probe for the sender-driven QoS and then select a PHB. Hence, the egress router (i.e. the new router joining the group), may use one of two types of join requests, an *absolute join* request or a *dynamic QoS join* request.

3.3.1.1 Absolute Join

The *absolute join* process operates using a simple request/response message exchange. When an egress router wishes to attach itself to the multicast tree, it tunnels a *Join-Request* message towards the source of the multicast group. The ingress router intercepts the *Join-Request* message and process the message. The *Join-Request* message includes the original inter-domain join request, the IP address of the new egress point, and the requested PHB for traffic to the new egress point.

Upon receipt of a *Join-Request* message, the ingress router may need to perform admission control by contacting the BB (Bandwidth Broker) of the domain if one is present. Once the new resource allocation has been approved, the ingress router updates the multicast tree by computing a new TEH and updating its shaping/policing mechanisms if necessary. The ingress router responds with a *Join-Ack* message to the new egress router. From this point onwards, all subsequent data packets sent out from the ingress router will be distributed to the new egress router.

3.3.1.2 Dynamic QoS Join

An alternative approach is for the new egress router to select a PHB based on network dynamics rather than specifying an absolute PHB. This approach is especially applicable for dynamic scheduling schemes such as relative DiffServ [87] or for adapting to other reservation mechanisms such as RSVP [26]. In the *dynamic QoS join* case, the egress point first tunnels a *Bid-Request* message towards the source of the multicast group. Similar to the absolute join, the ingress router intercepts the *Bid-Request* message and processes the message. In this case, the ingress router responds with a special *Bid-Probe* message towards the egress router. The

packet is routing explicitly using the DSMCast TEH using a multicast control group address.

As the packet crosses the domain, various statistics such as average queue size, average queue delay, and average loss rate are included for a fixed subset of classes. This scheme fits well within the DiffServ notion of a sender-driven QoS giving the egress router an approximate picture of the various performances of each class on the paths the actual data packets would travel on. Once the egress router receives the *Bid-Probe* message, it selects a PHB to join to the group with and follows the absolute join protocol. Note that this procedure may be invoked periodically to offer adaptive QoS.

Figure 3.7 summarizes the dynamic QoS join routine. In the first step, E_{RCVR} tunnels a *Bid-Request* message towards S. The *Bid-Request* message is routed via traditional IP routing and is intercepted by E_3. E_3 processes the message and replies with a *Bid-Probe* message back to E_{RCVR}. As the *Bid-Probe* message travels back to E_{RCVR}, various stats for classes are gathered at each core node along the path. The figure denotes only one of the pieces of core information that could be gathered. Upon receiving the *Bid-Probe* message, E_{RCVR} would dispatch a *Join-Request* message to E_3 since it now knows the ingress node for the multicast group. E_3 would then add E_{RCVR} to the multicast group and respond to E_{RCVR} with a *Join-Ack*. From that point onwards, new multicast data packets to the group would also flow to E_{RCVR}.

Although this approach offers users additional flexibility, the flexibility comes at an additional cost. First and foremost, the service requires additional complexity at core routers. Second, the burstiness of traffic may reduce the utility of the results if the results are not weighted properly. Third, the approach increases the join time and may be best applied after an initial join is complete. Finally, such a scheme would need to be rate-limited to prevent unnecessary consumption of router CPU resources.

3.3.2 Egress Join - Traditional IP Multicast

As stated earlier, the location vagueness of traditional IP multicast prevents an egress router from knowing the location of a multicast group. Thus, an egress router cannot simply

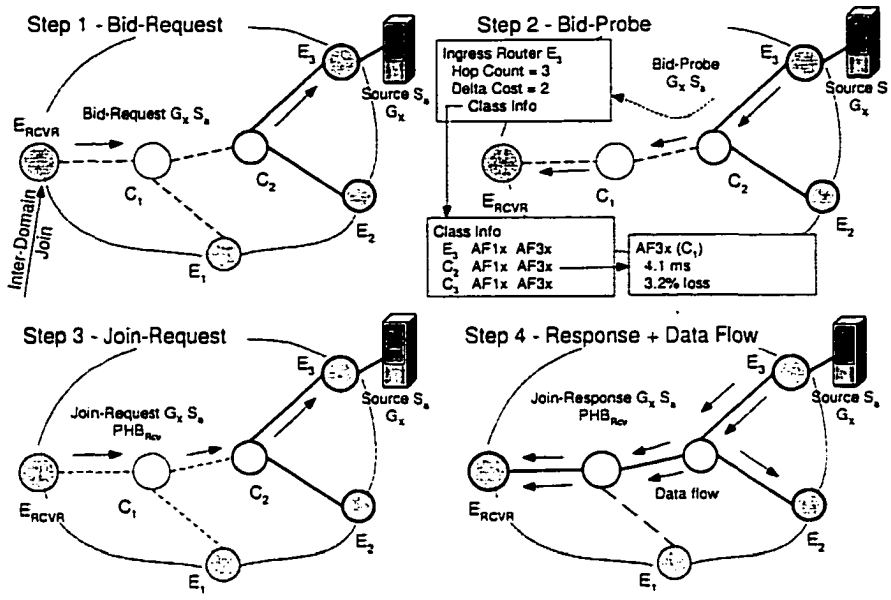


Figure 3.7 DSMCast join protocol - dynamic join - SSM

tunnel towards a source since there is not a source correlated to an traditional IP multicast group. The proposed solution for DSMCast is to simply expand the bid/probe routines of DSMCast.

When a new egress router to receive traffic for a multicast group, the egress router asks all of the potential edge routers using a *Bid-Request* message. Rather than sending out separate tunnels, the egress router multicasts via a special control tree that is a static multicast tree containing all eligible edge routers. Upon receiving a *Bid-Request* message, each edge router responds with a *Bid-Probe* message towards the new egress router. Depending upon the number of edge routers responding, the probe (information gathering) function of the *Bid-Probe* message may be disabled in the case of traditional IP multicast. If necessary, the edge routers may need to invoke inter-domain routing in order to locate the multicast group. The *Bid-Probe* message is not sent until the edge router has located the multicast group.

Since any edge router could be a potential candidate for the ingress router, the cost of the tree must be included as well. For instance, in a large domain it may be desirable to choose an alternative ingress router even though the multicast group is already present in the domain

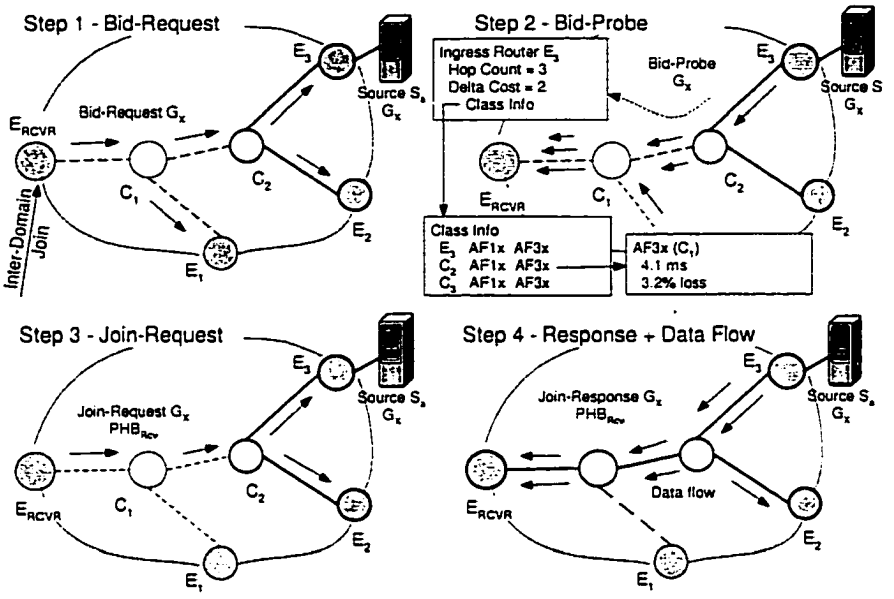


Figure 3.8 DSMCast join protocol - traditional IP multicast

at another router. At the new egress router, the various *Bid-Probe* messages are collected and processed until either a timer has expired and/or a sufficient threshold of edge routers have responded. Since the traditional IP multicast definition allows for multi-sourced groups (i.e. multiple ingress routers), the egress router will send a *Join-Request* to all edge routers that responded as being part of the multicast group. The *Join-Request* message may be conveyed using a subset of the control tree. In addition, the egress router may choose not to request an acknowledgement from each of the on-tree edge routers, thus reducing the control message overhead. To leave the traditional IP multicast group, a *Leave-Request* message should be multicast to the other on-tree edge routers.

Figure 3.8 shows an example of the traditional IP multicast extensions for DSMCast. Similar to the source-specific case, a *Bid-Request* message is sent in order to join the group. However, unlike in the previous case, the message is sent to all of the eligible edge nodes rather than the single ingress node. Each of the edge nodes responds and E_{RCVR} selects the best node to join to. Once E_{RCVR} has selected an ingress node, it will send a *Join-Request* message to that specific ingress node. Once the ingress node has received the *Join-Request* message, it

will respond with a *Join-Ack* message and future multicast data packets will also be sent to E_{RCVR}.

3.3.2.1 Shared Trees

One of the variations of multi-sourced groups in traditional IP multicast is the concept of a shared tree. Rather than each source having its own group address (known as many-to-many multicasting), a single group address is used to offer a consistent QoS or to share bandwidth for intra-group communications. Examples of such applications that might utilize shared trees include audio-conferencing where only one individual is speaking at a time.

From a routing perspective, the goal of the shared tree is to minimize the total consumed bandwidth while still satisfying the QoS of all receivers. In contrast, the shortest path tree prioritizes QoS by using the shortest path at the potential detriment to tree cost. Although shared trees are an interesting theoretical concept, the general trend is away from the complexities of shared trees and traditional IP multicast. Hence, explicit support for working with globally shared trees is not provided. However, the large body of research on shared tree construction and maintenance can still be applied on a domain-wise basis for optimizing the domain distribution tree [88].

3.4 Other Issues with DSMCast

With any new architecture for the Internet, the issues for deployment must also be examined as well as any future concerns. Several of the issues that must be considered with DSMCast include IPSec, initial deployment requirements, and IPv6. In addition, this section also considers other related work, enhancements to DSMCast, and implications beyond multicasting.

3.4.1 IPSec

The use of IPSec [9] does not present a problem with normal DSMCast operation and DSMCast with tunneling since all information related to the DSMCast header is removed at

the edge of the DS domain. However, the use of IPSec and DSMCast can cause problems when DSMCast is used with an adaptive DS field. In its default mode. IPSec does not include the DS field in its cryptographic calculation. Thus, the default mode of IPSec does not conflict with the use of heterogeneous QoS. However, the IPSec tunnel mode does encrypt the IP header. thus representing a problem with heterogeneous QoS. This problem is similar to the dilemma faced by inter-domain edge routers that do packet remarking [27].

3.4.2 Initial Deployment Requirements and IPv6

In order to deploy DSMCast for a given DS domain, all routers in the domain must be DSMCast-enabled. Although it would be possible to introduce a tunnel extension to DSMCast [73], the cost of such support is quite high. An intermediate step for DSMCast is to offer software support until hardware support is available or to employ an edge-only solution (see Chapter 4).

Since DSMCast is entirely transparent to IP routing (besides the changes to the DS field for variable QoS), DSMCast will not have any significant inter-operability issues with IPv6. The only potential issues lie in the unique ID field of DSMCast and the possibility for variable headers in IPv6. Whereas the last 8 bits of the IPv4 address are a viable approach. such may not be the case for IPv6. Due to the fact that the MAC address may simply become the last 6 bytes of the IPv6 address, it is quite likely that conflicts will emerge. As a result, it may be necessary for the network administrator to use a separate unique ID and to provide a mapping table to all routers in the network. The second potential issue lies with the possibility of multiple headers IPv6 headers. In such a case, a CAM or FPGA approach may be more difficult to use and thus, additional information may need to be included in the DSMCast header to optimize the search time in a software-based network processor.

3.4.3 Related Work to DSMCast

Small Group Multicast [89], a proposal under consideration by the IETF, uses a similar encapsulation-based technique to achieve traditional IP multicast. Under Small Group Mul-

ticast (SGM), the multicast tree is embedded inside the packet and sent from the source to a given SGM router. At the SGM router, the encapsulated tree information is appropriately partitioned and the new packet(s) are transmitted onto the downstream links.

DSMCast differs from Small Group Multicast in several key areas. First, DSMCast provides transparent support for traditional IP multicast across the DiffServ domain. Whereas SGM requires support at the source, DSMCast already works within the existing IP multicast architecture. Second, DSMCast core routers are significantly less complex than SGM-enabled routers. Under DSMCast, no modifications are made to packet length (except for encapsulation-based tunneling) thus requiring core routers to only replicate DSMCast packets and make only minor changes to the contents of the packets. In contrast, the SGM header changes at each router due to partitioning according to the addresses encapsulated within the packet. Although partitioning may reduce the downstream bandwidth, it adds a significant computation cost since SGM routers must partition and modify the size and contents of both the SGM and IP headers of packets.

Third, the DSMCast architecture takes special advantage of the underlying DiffServ architecture in order to provide both group size and group addresses (unique multicast IDs) scalability. Whereas SGM encapsulates the entire end-to-end multicast tree, DSMCast encapsulates only the multicast tree for a given DS domain. Thus, the use of SGM is limited to small groups since the IP addresses of all receivers must be included in the SGM header. In contrast, DSMCast is only restricted by the number of core routers in a multicast tree in a single DS domain, not by the end-to-end multicast tree nor by the number of receivers. Thus, the scalability of DSMCast is dependent upon the size of the domain, not the size of the multicast group which allows DSMCast to serve both large and small groups.

3.4.4 Enhancements to DSMCast

Although DSMCast is targeted towards multicasting, the use of DSMCast can have significant impacts in other areas including QoS management, unicast QoS, and fault detection.

3.4.4.1 QoS Management

To start, the *Bid-Request/Bid-Probe* messages can be adopted for more than simply determining the appropriate PHB for the egress router for multicasting. The *Bid-Request* and *Bid-Probe* sequence can be used for adaptively meeting the QoS for unicast aggregations of traffic as well. *Bid-Probe* messages can be sent periodically to other edge routers, regardless of whether or not the *Bid-Probe* message was solicited or not. Edge routers could then use the information provided by the *Bid-Probe* messages to dynamically adapt their shaping/policing or even routing patterns of incoming packets. This approach allows edge routers to be proactive towards QoS rather than waiting for a QoS violation to occur and reacting to the violation.

3.4.4.2 Unicast QoS

The introduction of heterogeneous QoS for multicasting poses an interesting question: would heterogeneous QoS be useful for unicast connections? For instance, would it make sense to offer a near-EF (Expedited Forwarding) service instead of absolute AF or absolute EF services?

Suppose that a customer wants better QoS than the AF2x service the company is currently using. Rather than upgrading the traffic to AF1x, selected hops/links could be upgraded to AF1x. An example of such a case would be the traffic receiving AF1x-AF2x-AF2x-AF1x treatment instead of strictly AF1x or AF2x. A service provider would be able to offer a gradient of QoS rather than absolute QoS from edge-to-edge. For underlying schedulers such as relative DiffServ [87], the ability to adaptively manage PHBs over different links in the core offers even greater benefit.

The DSMCast architecture can be adapted for unicast connections by placing only the route to only one egress node in the TEH. Although such a scheme would offer greater flexibility, the cost would not be insignificant. The implications of variable QoS for unicast connections has the potential for dramatic implications in DiffServ and is an open topic for future research.

3.4.4.3 Fault Detection

The final aspect where DSMCast can offer additional benefit is in the area of fault detection. Although the route-pinned nature of DSMCast makes DSMCast slower to react to faults due to delays in network updates. the route-pinned nature of DSMCast allows edge routers to verify the health of link and routers themselves. Rather than relying on CPU intensive link state messages. the edge routers can assist the link state protocol while monitoring the QoS of the network. Chapter 6 discusses this aspect in significantly greater detail.

3.5 Theoretical and Simulation Studies

In this section, the performance of DSMCast is analyzed for both theoretical as well as simulated performance. In the theoretical analysis. DSMCast is examined for its overhead versus the edge-based approach. the state-based approach. and other factors such as network size and fragmentation. The simulation studies examine the performance of DSMCast versus the other two approaches (state-based and edge-based) regarding a variety of parameters including group size. packet size. receiver heterogeneity. and group dynamics.

3.5.1 Theoretical Studies

3.5.1.1 Examining the Cost of State in the Core

As mentioned earlier. the primary benefit of DSMCast is that it reduces the cost of multicast state information to make multicasting more scalable. The justification for the additional overhead of DSMCast is that the cost of maintaining state information is greater than the cost of the additional bandwidth imposed by the stateless approach. Formally, this relationship can be summarized as follows:

$$Cost_{State}[T_E + T_C] + Cost_{BW}[(T_E - 1 + T_C) \times P_S] >$$

$$Cost_{State}[T_E] + Cost_{BW}[(T_E - 1 + T_C)(P_S + OH)]$$

where T_E is the number of edge nodes in the tree, T_C is the number of core nodes in the tree, P_S is the size of packet, OH is the per-packet overhead introduced by stateless core routing, $Cost_{State}[]$ is a function denoting the cost of state, and $Cost_{BW}[]$ is a function denoting the cost of bandwidth. The equation can be reduced as follows:

$$Cost_{State}[T_C] > Cost_{BW}[(T_E - 1 + T_C) \times OH]$$

which essentially states that the cost of the state information in the core is greater than the per-packet overhead introduced on the entire multicast tree for the domain. This is a valid assumption for several reasons. First, the cost of bandwidth has dramatically decreased over the last few years and new advances are continually increasing the bandwidth available [38]. Second and most importantly, although multicast may or may not make up a large portion of traffic, the number of unique multicast groups (especially with SSM) will explode dramatically. Although a router may be able to manage the impact of millions of entries in the routing table, the management of the complex per-group state information (timers/etc.) will simply not be scalable. Thus, support for a stateless core is critical to the successful deployment and wide scale acceptance of multicasting beyond MBone [47].

3.5.1.2 Overhead of DSMCast

The overhead of the DSMCast TEH is dependent upon the number of non-leaf routers in the domain that are on the multicast tree. For each router where replication takes place, an entry will be present in the TEH. The cost of the TEH can be summarized below:

$$Cost = NumEntries + Options + k * EntrySize$$

where $NumEntries$ is the field giving the number of entries in the TEH (1 byte). $Options$ is the options field (1 byte), k is the number of entries, and $EntrySize$ is the size of each entry. The cost of each entry is given below:

$$EntrySize = \lceil \frac{ID + Rep + Rep * SizeQoS}{8} \rceil$$

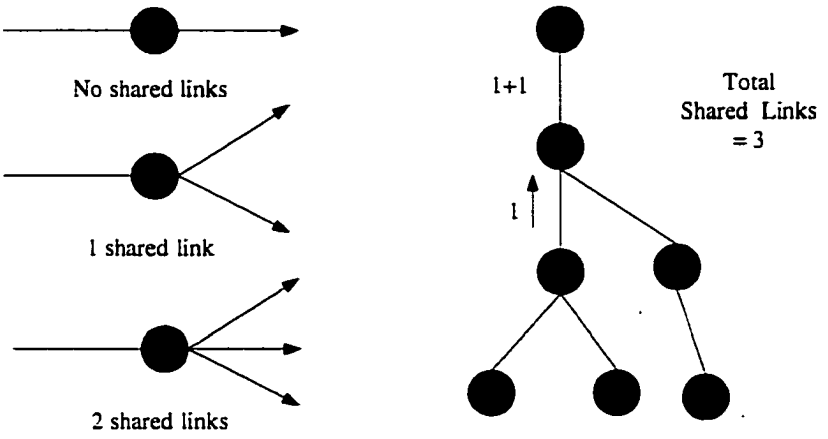


Figure 3.9 Definition of shared links

where ID is the size of the unique ID (8 or 16 bits), Rep is the maximum number of interfaces on any node in the domain (5 or 6 bits), and $SizeQoS$ is the size of the QoS transformation field (2 or 3 bits). In addition, each entry is forced to byte boundaries to ensure ease of hardware support. The 5/2 ($Rep = 5, SizeQoS = 2$) and 6/3 settings deliver sizes of 23 bits (rounded up to 3 bytes) and 32 bits (4 bytes) respectively.

3.5.1.3 Tradeoff versus the edge-based approach

In order for DSMCast to offer a significant benefit, it must offer a better performance than the simplest edge-based approach, ingress-branching. The ingress-branching solution allows only the ingress node to replicate packets and then such packets are tunneled to the appropriate egress routers. Although less efficient than DSMCast, this scheme satisfies the three conflicts between DiffServ and multicasting. In fact, the scheme on a global scale can still offer performance benefits as tree-wise aggregation still occurs at the edge of the domain. Thus, DSMCast must justify its additional complexity by offering a significant savings over this simple scheme.

In short, DSMCast offers performance a performance improvement when shared links occur in the multicast tree. A shared link occurs when more than one egress point shares a branch of the multicast tree (see Figure 3.9). The number of shared links at a given node (x) in the

multicast tree can be defined as:

$$SL_x = \sum_{i=0}^{C} SL_i$$

where C is the total number of children at the node and SL_i is the number of shared links at each node in the multicast tree. The total number of shared links at a given node is the aggregation of all of the shared links of its children. With each shared link, DSMCast saves the cost of an entire packet transmission versus the separate tunnels of the ingress-branching approach. However, whereas the ingress-branching approach incurs only a relatively small penalty for the tunneling overhead, the DSMCast overhead is significantly greater. The cost of the ingress-only approach can be summarized below:

$$Cost_{Ingress} = \sum_{x=0}^{R} D(E_{Ingress}, E_x) * (P_S + T)$$

where R is the number of egress points (receivers), $D(E_{Ingress}, E_x)$ is the distance (in hops) between $E_{Ingress}$ and E_x, P_S is the size of the packet, and T is the cost of tunneling. For the same tree, the cost of DSMCast is as follows:

$$Cost_{DSMCast} = (N-1) * (P_S + TEH)$$

where N is the total number of nodes in the tree. P_S is the size of the packet, and TEH is the size of the DSMCast Tree Encapsulation Header. The tradeoff between the two approaches can be captured by the number of shared links. The number of shared links is simply the total hops required by the ingress-branching approach minus the total links in the tree. The number of shared links is captured below:

$$\sum_{x=0}^{R} D(E_{Ingress}, E_x) - (N-1) = \sum_{i=0}^{N} SL_i$$

Hence, the tradeoff at which DSMCast offers a benefit can be captured as the following:

$$(N-1)(P_S + T) + \sum_{i=0}^{N} SL_i \times (P_S + T) \geq (N-1)(P_S + TEH)$$

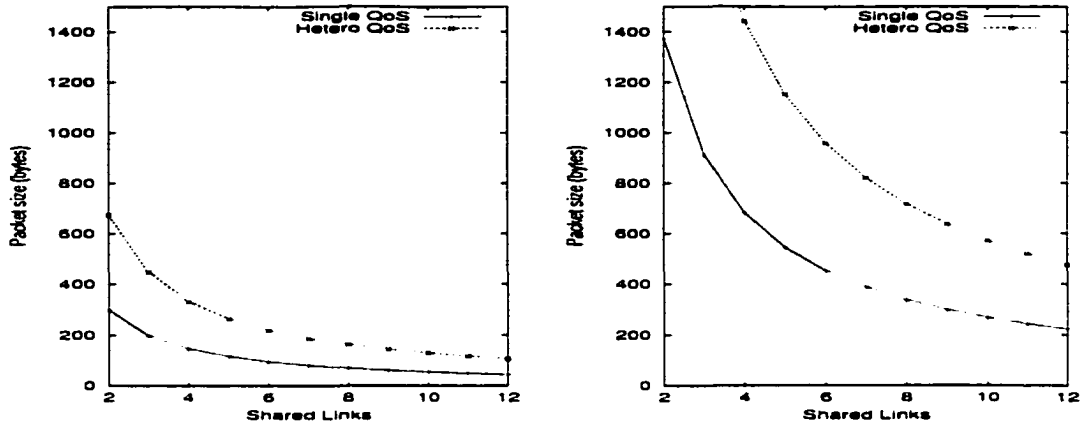


Figure 3.10 Tradeoff between ingress-branching and DSMCast (a) 32 core. 16 edge nodes (b) 64 core, 32 edge nodes

$$\sum_{i=0}^{N} SL_i \times (P_S + T) \geq (N-1)(TEH - T)$$

$$P_S \geq \frac{(N-1)(TEH-T)}{\sum_{i=0}^{N} SL_i} - T$$

where TEH is the size of the of DSMCast header (dependent upon the size of the tree), and T is the cost of tunneling (Minimal Encapsulation - 12 bytes [64]). The equation informally states that when the number of shared links is sufficient, the packet will be replicated enough additional times in the ingress-branching case to offset the cost of the Tree Encapsulation Header on the entirety of the multicast tree.

Figures 3.10(a) and 3.10(b) plot the tradeoff at which DSMCast offers a benefit versus ingress-branching. In both graphs, it is assumed that 50% of the edge nodes are part of the tree and that 50% of the core nodes are part of the tree as well. Although the tradeoff of 470 bytes may seem high for the network of 96 nodes (64 core + 32 edge nodes) and 12 shared links, consider the relatively low numbers of shared links that were plotted. For even the small example listed in Figure 3.9, there were already 3 shared links present for only 3 receivers. In

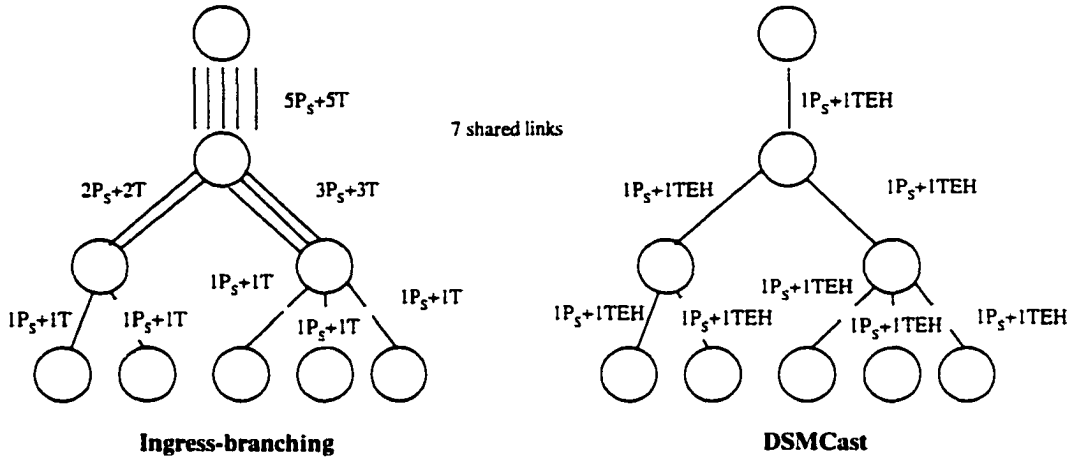


Figure 3.11 Definition of shared links

fact, the simulation studies in the next section frequently found shared links equal or greater than those plotted here

Figure 3.11 shows another example of calculating the number of shared links. With only 5 receivers, a fairly small network was able to achieve 7 shared links, thus making the necessary shared link tradeoff fairly easy to achieve. The figure also introduces an additional benefit of DSMCast versus ingress-branching. With DSMCast, a significant amount of congestion is relieved at upstream nodes near the ingress. Note that the impact of DSMCast on each link is constant whereas the impact of ingress-branching varies considerably depending upon the distance from the ingress router. However, despite the fact the oftentimes DSMCast should offer a better performance versus ingress-branching, there may be cases when DSMCast does not offer the best performance. In such a case, DSMCast will then switch to ingress-branching to optimize the bandwidth consumed but still maintain the benefits of core statelessness.

3.5.1.4 Tradeoff versus state-based

Figures 3.12(a) and 3.12(b) show the overhead of DSMCast normalized to the performance of traditional IP multicast (state-based approach) on two different network sizes. Once DSMCast passes a sufficient packet size, the costs of DSMCast become negligible. In fact, without

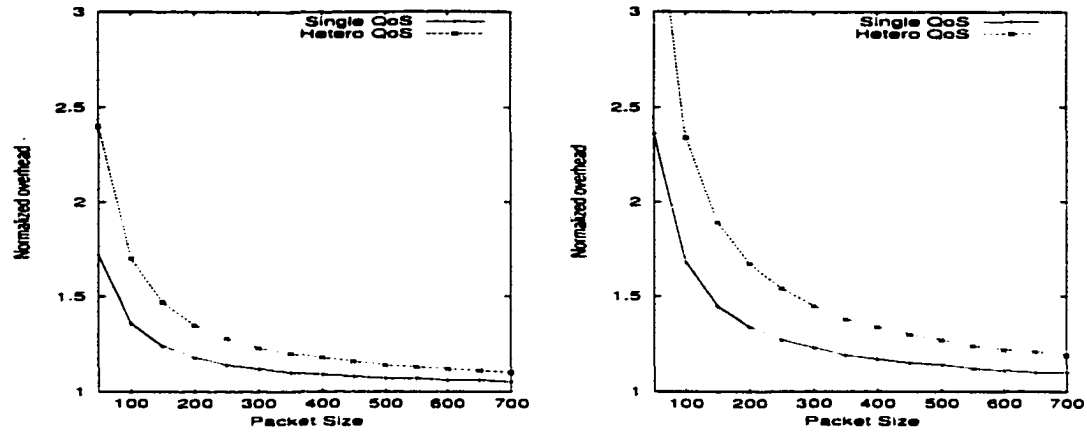


Figure 3.12 Normalized overhead versus state-based for DSMCast with (a) 32 core. 16 edge nodes (b) 64 core, 32 edge nodes

variable QoS enabled, the costs of DSMCast is reduced even further to the point where the overhead introduced by DSMCast would be less than the performance penalty of enabling state-based IP multicast on many multicast routers.

3.5.1.5 Effect of Fragmentation

One of the undesirable side effects of DSMCast is the potential for additional packet fragmentation. If the DSMCast TEH causes the packet size to exceed the MTU (Maximum Transmission Unit) of the network, the packet must be fragmented into two separate packets. As a result, the newly fragmented packet would incur the additional bandwidth overhead of a new IP header and an additional DSMCast TEH as well as additional processing at both the ingress router as well as the final receiver. In addition, the chance of losing the packet increases due to the fact that the loss of either fragment would cause the entire packet to be lost. The cost of DSMCast with fragmentation is specified below:

$$Cost_{DSMCast} = (N-1)(\frac{P_S}{2} + TEH) + (N-1)(\frac{P_S}{2} + TEH + IP)$$

Figures 3.13(a) and 3.13(b) show the effects of fragmentation on the normalized overhead

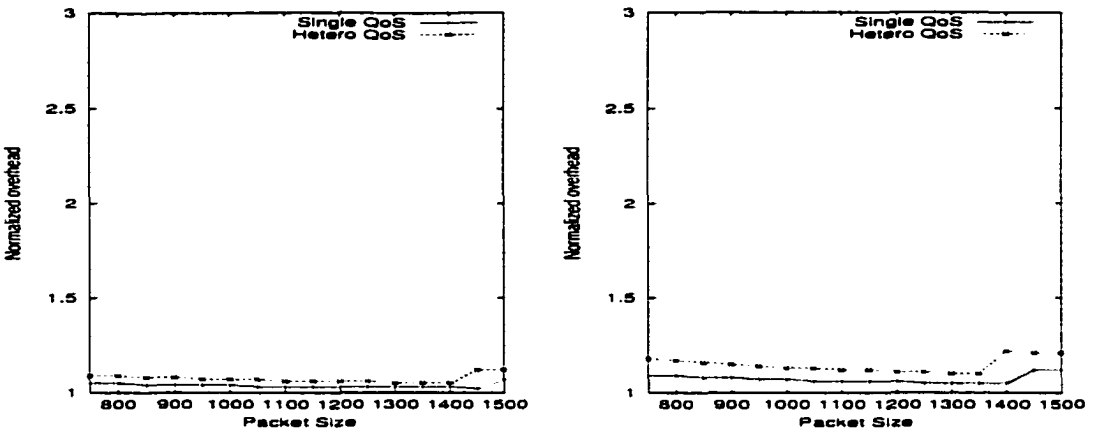


Figure 3.13 Normalized overhead including fragmentation versus state-based for DSMCast with (a) 32 core. 16 edge nodes (b) 64 core. 32 edge nodes

of DSMCast. The MTU of Ethernet of 1500 bytes is used with the same parameters as Figure 3.12.

3.6 Simulation Studies

The performance of DSMCast versus other solutions was analyzed through extensive simulation studies. The simulation studies were conducted using the *ns-2* simulator [90] and the GenMCast extension for multicasting [91]. The performance of DSMCast was compared to the performance of the ingress-branching solution and a generic version of PIM-SM [69]. The parameters for the simulations were as follows:

- The network topology consisted of 32 core nodes connected with an average node degree of 3. 16 edge routers connected by at least one core router in between two edge routers. NSFNet was also used as a realistic topology for comparison [92].
- Each edge router was the source for two SSM multicast groups (32 total groups).
- Each multicast group produced CBR traffic with an exponentially distributed packet size

Table 3.3 DSMCast ns parameter list

Parameter	Value(s)
Link Bandwidth/Delay	1.0 Gb/s, 5 ms
PHBs	AFxy (x=1..3,y=1..4)
Uniform	x(25%,25%,25%,25%),y(33%.33%,33%)
Non-Uniform	(10%,10%,40%,40%),(10%,30%.60%)

and packet inter-arrival time. Unless otherwise noted, the mean for the values were 400 bytes and 100 ms.

- The starting number of egress points for each simulation was 160 egress points (average group density of $\frac{160}{32} = 5$ egress points) with the egress points randomly distributed between the various groups.

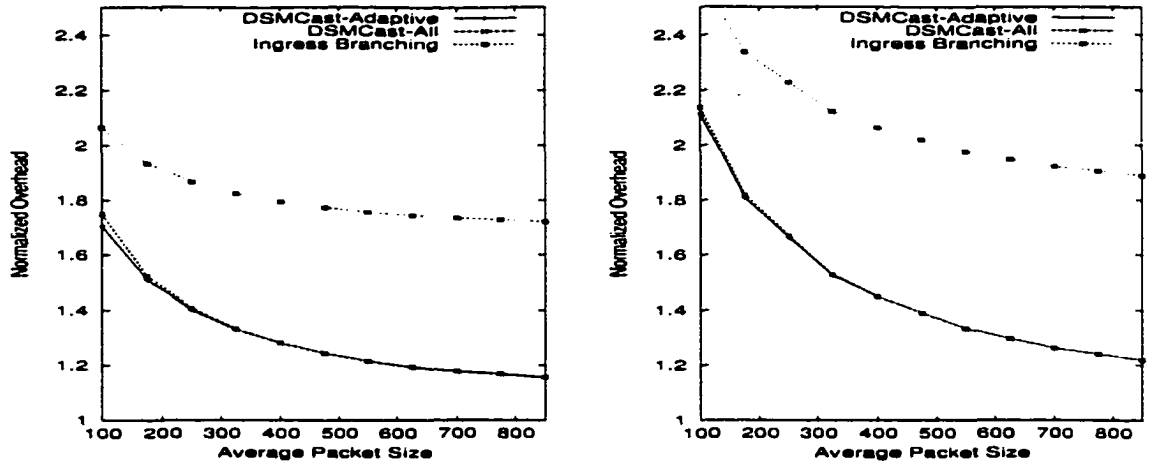
- The events that constitute group dynamics (join/leave) were uniformly distributed with a mean inter-event time ranging from 150 ms to 250 ms. The probability for a join or leave was 0.5 for each.

- The network was assumed to have sufficient buffer space and bandwidth such that packet dropping does not occur. This is done to ensure an adequate comparison between the various solutions without bias by packet drops.

- Additional parameters are summarized in Table 3.3.

The simulations were conducted for a single domain scenario of varying random network topologies and varying QoS distributions (uniform and non-uniform). The models were evaluated on basis of the normalized cost versus the state-based approach (PIM). The normalized overhead gives an indication of the relative bandwidth cost of the model in a loss-free network. For example, an overhead of 1.2 denotes that an algorithm requires 20% extra bandwidth. For each of the figures, the following values are graphed:

- *PIM-SM:* A generic version of PIM-SM is used as a benchmark for the simulations. The results of the simulations are normalized to the cost of PIM-SM.



(a) (b)

Figure 3.14 Effect of packet size on overhead - uniform distribution - normalized - (a) Random network (b) NSFNet

- *Ingress-branching:* The ingress-branching approach only replicates packets at the ingress of the domain. Packets are then tunneled across the domain using Minimal Encapsulation [64].

- *DSMCast-All:* The DSMCast-only version relies exclusively on DSMCast regardless of packet size. Hence, if the packet size is too low, the performance may be worse than the ingress-branching solution.

- *DSMCast-Adaptive:* The DSMCast-adaptive version alternates between either DSMCast or ingress-branching depending upon the optimal solution. At the worst case, the performance of this solution should be the same as the ingress-branching solution.

3.6.1 Effect of Packet Size

Figure 3.14 shows the performance of the three schemes as the mean packet size is varied from 100 to 850 bytes. As the average packet size increases, the overhead of DSMCast decreases due to the fact that the overhead of DSMCast becomes a smaller percentage of the overall packet size. This assertion is supported by the results shown in Figure 3.15. Figure 3.15 shows

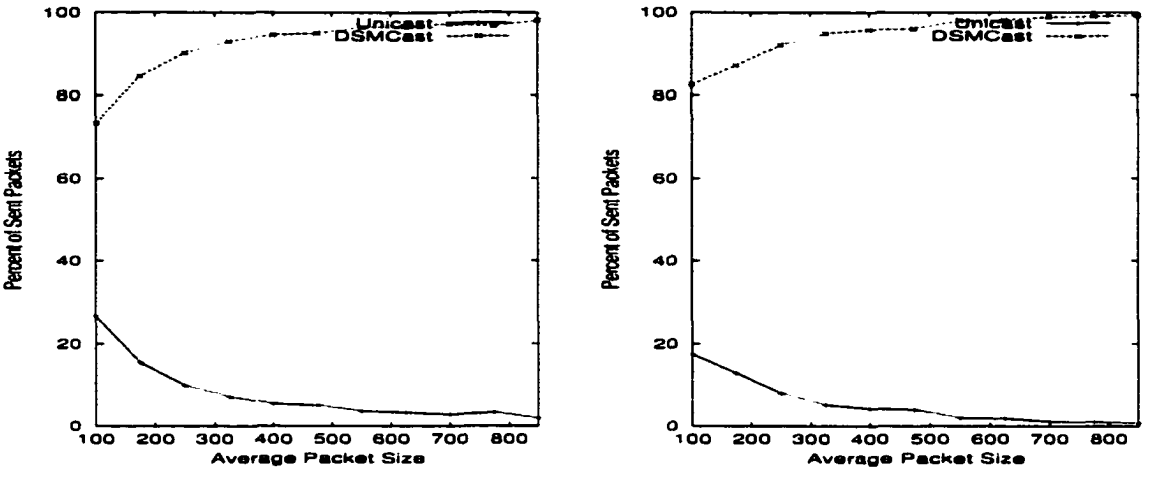


Figure 3.15 Effect of packet size on DSMCast adaptive selection - uniform distribution - normalized - (a) Random network (b) NSFNet

the selection choices of the adaptive DSMCast model. As the packet size increases, the tradeoff point of DSMCast falls, thus causing the adaptive algorithm to choose DSMCast more than separately unicasting the packets (ingress tunneling). In fact, even at a relatively low average packet size (100 bytes), the tradeoff for DSMCast is sufficiently small enough to justify over 80% of the packets being sent by DSMCast. Furthermore, the multimedia nature of most group-based QoS applications typically have packet sizes well beyond those simulated, thus reducing the encapsulation overhead to well below 15% more than the state-based approach.

3.6.2 Effect of Group Size

Figure 3.16 examines the performance of DSMCast further as the number of egress points is varied. An increase in the number of egress points implies that additional downstream receivers wish to subscribe to the multicast group. The number of egress points are kept fairly sparse, ranging from less than 25% of the total egress points (average 3 egress points per group) to 50% of the total egress points (average 8 egress points per group). Whereas the overhead of DSMCast does rise, it can be directly attributed to the additional links that are present in the tree.

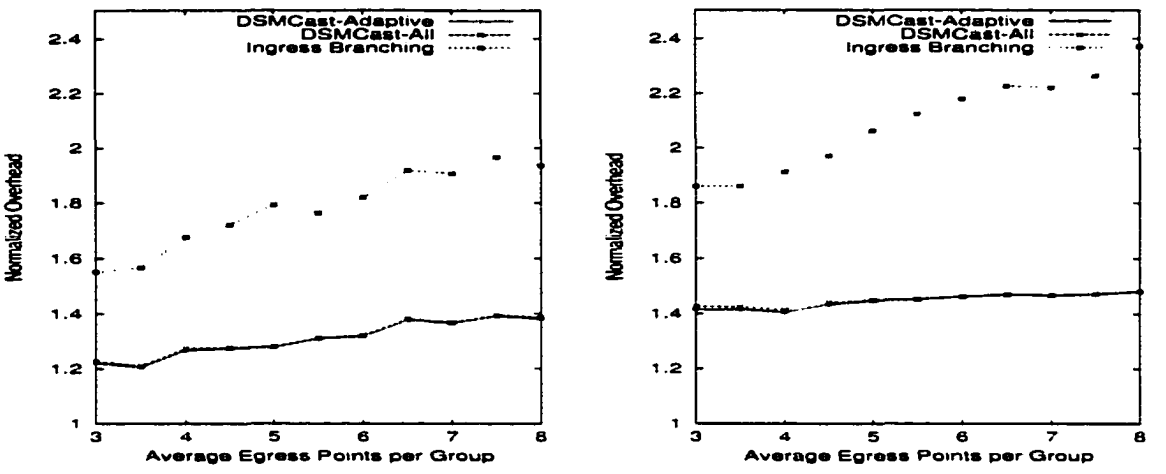


Figure 3.16 Effect of group size on overhead - uniform distribution - normalized - (a) Random network (b) NSFNet

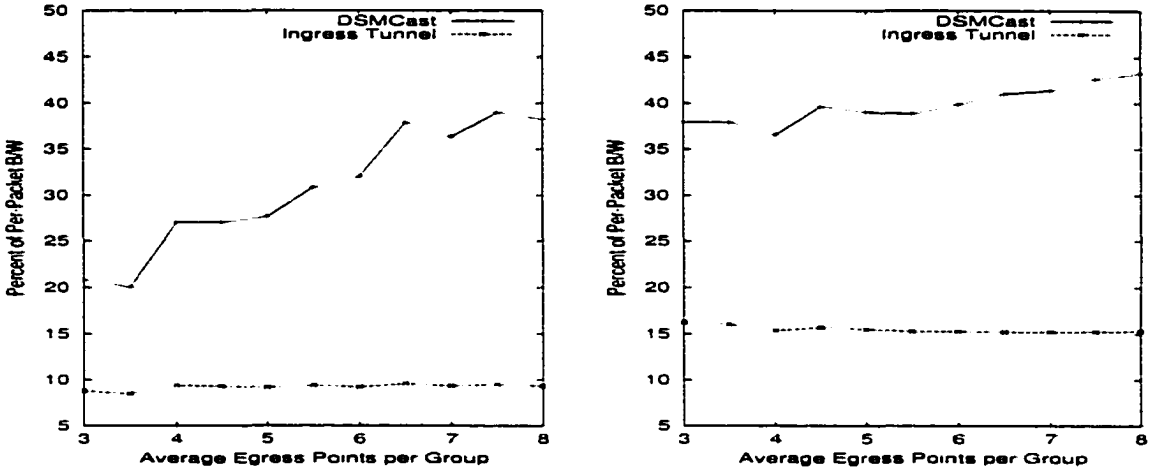
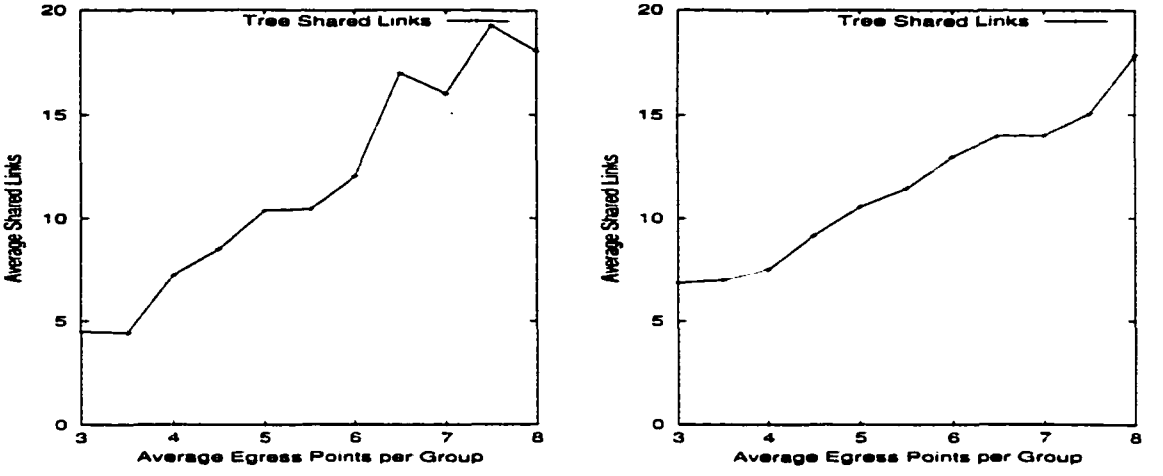


Figure 3.17 Effect of group size on packet header size - uniform distribution - normalized - (a) Random network (b) NSFNet



(a) (b)

Figure 3.18 Effect of group size on DSMCast adaptive selection - uniform distribution - normalized - (a) Random network (b) NSFNet

Figure 3.17 breaks down the cost of the per-packet overhead for both the ingress-branching (fixed tunnel cost [64]) and DSMCast. The percentage represents the overhead as a percentage of the data packet size used by PIM. Due to the nature of the random network, each additional egress point may potentially increase the number of core nodes in the tree, hence increasing the size of the DSMCast TEH. For a sparse network such as NSFNet, the increase in egress points barely affects the overhead of DSMCast.

In contrast, the introduction of additional egress points does offer several benefits to DSMCast versus ingress-branching. As the number of egress points increases, the chance for additional shared links increases as well, thus driving down the tradeoff for when DSMCast becomes practical. Figure 3.18 plots the increase in shared links as the number of egress points increases. The number of shared links increases faster than the number of egress points as a single egress point will typically utilize several shared links. In fact, as the group density increases, shared links will increase at a faster rate due to the presence of other egress points already being on the multicast tree. The increase in shared links produces a noticeable effect on the adaptive DSMCast model as the additional shared links reduce the tradeoff and hence, cause the adaptive model to select DSMCast more frequently (see Figure 3.19).

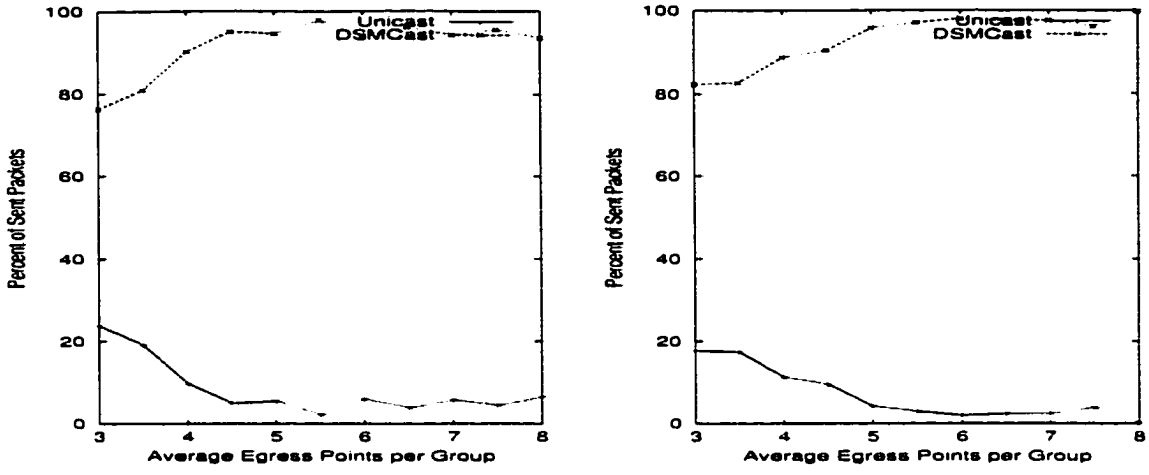


Figure 3.19 Effect of group size on DSMCast adaptive selection - uniform distribution - normalized - (a) Random network (b) NSFNet

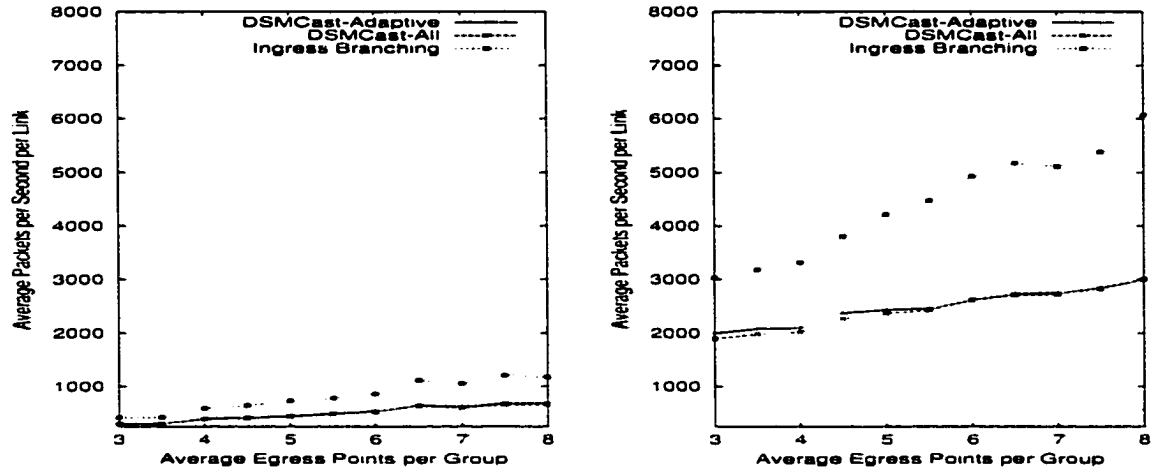


Figure 3.20 Effect of group size on the number of transmitted packets - uniform distribution - normalized - (a) Random network (b) NSFNet

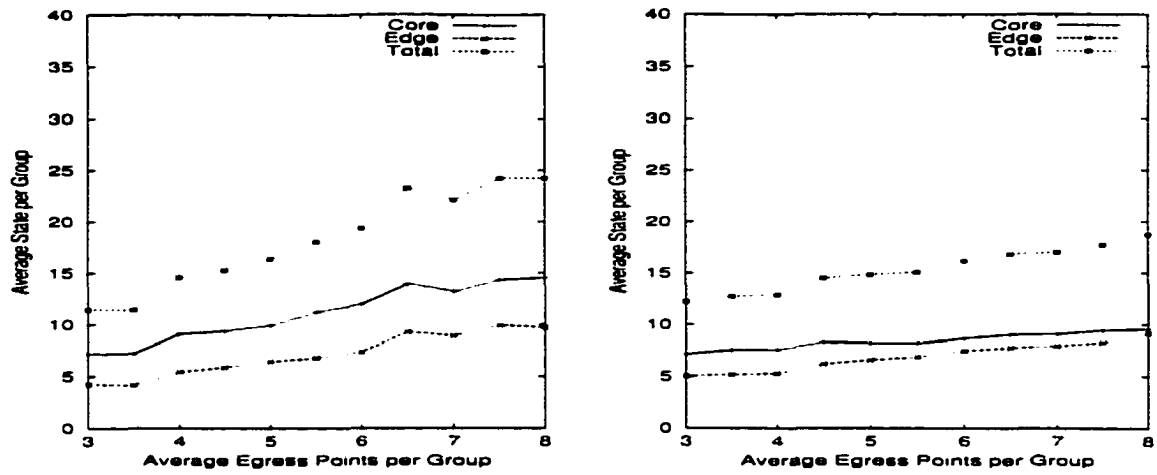
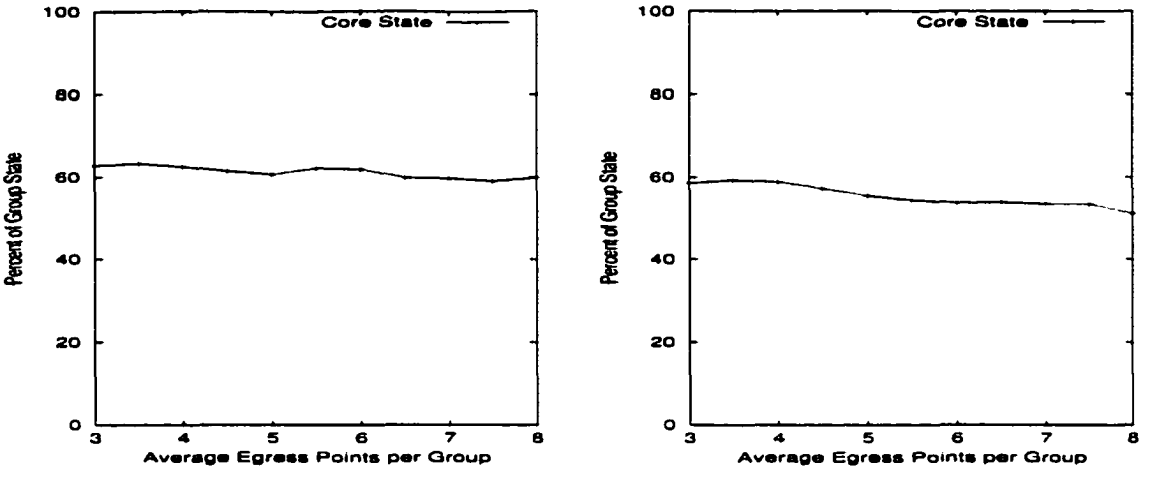


Figure 3.21 Effect of group size on state cost - uniform distribution - normalized - (a) Random network (b) NSFNet

A second way to analyze the tradeoff between ingress-branching and DSMCast is to examine the number of packets transmitted onto the network. Whereas both DSMCast and PIM minimize the number of packets due to the nature of the multicast tree. the ingress-branching method introduces one packet per egress point. As a result. the number of packets dramatically increases versus DSMCast and PIM. Figure 3.20 plots the average number of multicast packets per second transmitted onto each link in the entire network. The number of packets employed by PIM is the same as the number of packets employed by the non-adaptive version of DSMCast (DSMCast-All). The increase in packets can be directly tied to the increase in shared links as show earlier in Figure 3.18.

3.6.3 Overhead of the State-based Approach

While the earlier figures showed the performance of DSMCast relative to PIM and traditional IP multicasting, Figure 3.21 and 3.22 examine the state cost associated with the previous simulations. Whereas the sparse core of NSFNet allows the number of edge nodes to close the gap versus the number of edge nodes, the random network does not offer the same benefit. However, in both network settings, the amount of state information in the network represents



(a) (b)

Figure 3.22 Effect of group size on core state cost - uniform distribution - normalized - (a) Random network (b) NSFNet

over 50% of the total state information supported by the network. In fact, unless a sizeable amount of egress points are achieved in the group, the core state information is likely to dominate the memory/storage/maintenance requirements for multicast routers in the domain.

3.6.4 Effect of Group Size - Non-Uniform PHBs

Figure 3.23 plots the performance of the models as non-uniform traffic is applied. The performance versus the original uniform traffic model does not change as the size of DSMCast is dependent only upon the number of nodes in the tree, not the QoS requested by the egress points. Although cases may arise where the QoS transformation code may be insufficient to meet the needs of the requested QoS, the impact on the overhead of DSMCast would be negligible.

3.7 Conclusions

In this chapter, the DSMCast architecture was proposed and analyzed through both simulation and theoretical studies. Despite the fact that DSMCast incurs per-packet overhead due

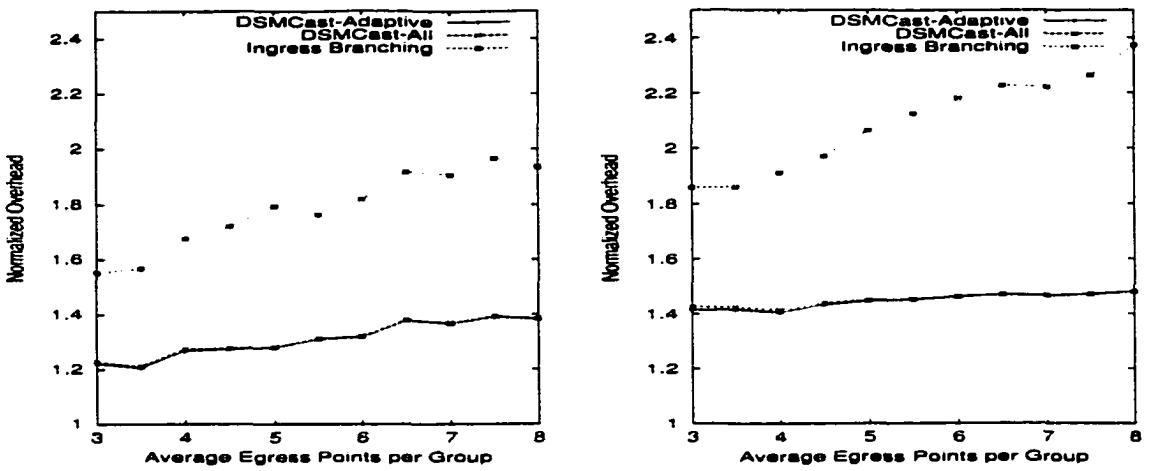


Figure 3.23 Effect of group size on overhead - non-uniform distribution - normalized - (a) Random network (b) NSFNet

to the encapsulation header, it is able to operate reasonably within the cost of the state-based approach. In addition, DSMCast can offer the performance while dramatically reducing the state requirements of the network. Although DSMCast does need hardware acceleration to avoid impacting the router CPU, the performance benefits of multicasting and the reduction of state information more than offset the cost of the hardware. Unlike simply extending traditional IP multicasting, DSMCast satisfies the three conflicts between DiffServ multicasting while incurring a negligible overhead. Thus, DSMCast can offer a viable approach for DiffServ multicasting.

CHAPTER 4. EBM: STATELESSNESS THROUGH TUNNELING

In this chapter, the second architecture for DiffServ multicasting, EBM (Edge-Based Multicasting), is proposed. Similar to DSMCast, EBM reduces the DiffServ multicast problem to edge-to-edge transport across a single DiffServ domain. However, EBM employs an edge-based approach that relies on standard IP routing and tunneling rather than route-pinning and hardware acceleration. The EBM architecture incorporates the concept of a Multicast Broker (MB) for group management and a novel algorithm for tree construction, the Edge Cluster Tree (ECT), that addresses the unique characteristics of heterogeneous QoS in the DiffServ multicasting environment.

The chapter is organized as follows. First, Section 4.1 proposes the EBM architecture, the concept of a Multicast Broker (MB), the egress join/leave protocol, and a simple tree construction algorithm. Next, Section 4.2 describes the ECT algorithm and presents a detailed example of ECT operation. Then, Section 4.3 examines the benefits of the EBM architecture in greater detail and contrasts EBM versus existing solutions and DSMCast. Section 4.4 investigates the performance of EBM versus the state-based and encapsulation-based approaches through extensive simulation studies. Finally, Section 4.5 offers several concluding remarks regarding the EBM architecture.

4.1 EBM (Edge-based Multicasting) Architecture

The primary goal of the Edge-Based Multicasting (EBM) architecture is to provide a multicast transport across a single DiffServ (DS) domain that addresses the three conflicts between DiffServ and multicasting. The first step is to apply the scalability principles of DiffServ to multicasting, i.e. push the state to the edge of the domain. The concept of edge-only in-

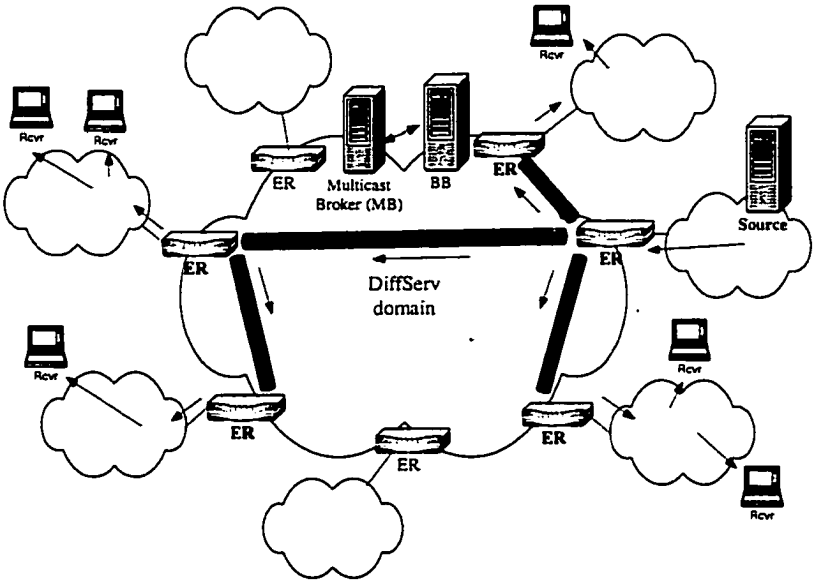


Figure 4.1 EBM network architecture

telligence fits perfectly with the notion of a multicast transport service that allows the core routers to remain as simple, stateless routers. Figure 4.1 shows the EBM architecture and the principles of EBM are summarized below:

- *Stateless core:* Core routers are multicast unaware and therefore do not maintain any multicast state information.
- *Tunneled packets:* Packets are tunneled from edge to edge, thus reducing the multicast packet to a true unicast packet in the core.
- *Edge replication:* Packets may only be replicated at edge routers. The replication information may be included in the packet (encapsulated tree) or maintained as state information at the edge router.
- *Multicast Broker:* The Multicast Broker (MB) manages the egress join/leave requests and multicast trees for the DS domain or individual groups at an ingress router. In addition, the multicast broker manages QoS interactions for multicasting (security, re-

source reservations, etc.). The MB may be a centralized (traditional IP multicast) or distributed (SSM) entity.

4.1.1 EBM Multicast Transport

In order to provide a multicast transport service, the edge router must provide two new functions, namely packet replication/tunneling and join/leave forwarding. Both functions are fairly straightforward and require only minimal changes to the edge router.

For packet replication and tunneling functionality, an edge router must be able to recognize a multicast packet and appropriately replicate/tunnel the packet onto the domain. Multicast packets may arrive using a multicast destination address (originated outside the domain) or as a tunneled packet. When a multicast packet arrives at an edge router from outside the domain, the edge router will examine its state information (provided by interactions with the MB) and replicate/tunnel the packet via a modified Minimal Encapsulation header [64] to downstream egress points in the multicast domain. For packets arriving at the edge router that are tunneled with the modified Minimal Encapsulation header, the edge router may look either at its own state information or inside the packet (encapsulated tree) to determine how to replicate the packet. If the packet should be replicated to outside of the DS domain, the encapsulation header (if present) is stripped off and the original information replaced in the multicast packet.

The second function of EBM, join/leave forwarding, involves the edge router acting as an attendant for inter-domain routing requests that require the intervention of the MB. For any join/leave request, the edge router must tunnel the request to the MB along with information about the QoS requested (PHB) and the address of the egress router requesting the multicast service. The MB processes the join/leave request and updates the appropriate edge routers with the new tree information. Thus, the edge router need not be concerned with the specifics of multicast routing, only the appropriate mechanisms for packet tunneling, replication, and forwarding to the MB.

4.1.2 EBM Multicast Broker (MB)

In order to simplify group management, the EBM architecture incorporates the use of an entity known as the Multicast Broker (MB). The responsibilities of the MB include tree construction, tree rearrangement, and security management. The MB may be either a *centralized* or *distributed* entity. If the MB is centralized, it is considered to be near (but separate) from the Bandwidth Broker (BB). For groups employing traditional IP multicast $(*, G)$, a centralized MB for the domain is employed. However, for groups where the source is known such as SSM (S, G), the MB may either be centralized or distributed (present at the ingress router). Note that distributed does not refer to the computational load, rather only the locations where the different pieces of group information are kept for the domain.

4.1.2.1 Centralized vs. Distributed MB

The centralization of all multicast information offers several significant benefits. First, QoS negotiations for a tree are greatly simplified. Due to the distributed nature of routing in traditional IP multicasting, each router may need to negotiate resources individually [76]. In contrast, the MB can negotiate with the BB for the entire multicast tree in one exchange. Second, the centralized approach lends itself towards a more robust security scheme as well. In contrast to traditional IP multicasting, the centralized approach makes it easier to force security features (authentication/encryption) on all group dynamics (topology changes, etc.) and can also serve as a starting point for managing secure groups.

Finally, and most importantly, the centralized architecture is ideal for tree construction and tree rearrangement [88]. Rather than acting on partial knowledge of the tree, a centralized architecture allows the MB to optimize the tree using more efficient centralized algorithms. In addition, the centralized architecture is best suited to consider the QoS impacts of tree changes (jitter, out-of-order delivery, etc.) when rearranging the domain multicast tree.

The primary disadvantage to a centralized MB is that it becomes a single point of failure. This disadvantage may be offset by approaches such as hot-cold standby or a primary/backup. In addition, the centralized MB may introduce additional join/leave latency that would not be

present otherwise. The location of the MB may also introduce bandwidth bottlenecks if the volume of group dynamics (join/leave) is exceptionally high with a centralized MB. Unless the ingress routers cannot handle the complexities associated with tree construction, a centralized MB should be used for traditional IP multicast groups and a distributed MB (ingress routers acting as MBs for individual groups) should be used for source-specific groups.

4.1.3 EBM Egress Join

One of the most critical operations of multicasting is the egress (member) join operation. The egress join operation determines the QoS that the downstream members of the multicast group will receive. In EBM, a member is defined as an egress point (edge router) that wishes to receive packets for the multicast group. An egress point may have many other downstream domains or receivers. The maximum size of an EBM multicast group is bounded by the number of edge routers (potential egress points) and is independent of the number of receivers, hence reducing the practical size of the problem.

The egress join operation can be summarized in three main steps:

- *Join* - The inter-domain join request is received and forwarded to MB.
- *MB processing* - The MB (ingress or centralized) locates the multicast group and calculates the new tree.
- *Graft* - The MB grafts the new edge router onto the tree.

Initial Join Request: The join process begins when an inter-domain join request reaches the edge of the DS domain. The request may be sent by any of the existing multicast protocols (SSM, IGMP, PIM) and would be intercepted by an edge router. Upon receiving a join request at an edge router ER_{NEW}, the request is forwarded to the MB for the group (known via configuration, discovery, or from the join request). In the event that MB of the group is the ingress router and the address of ingress router is not known, the packet is simply tunneled towards the source of the group and intercepted by the ingress router (the MB of the group). The request is forwarded to the MB using either a signaling PHB or other appropriate low

loss/reduced delay PHB. The *EBM-Join-Request* message includes the IP address of the new egress point (ER_{NEW}), the requested PHB (if possible), and the information from the inter-domain join request.

MB Processing: Upon receiving an *EBM-Join-Request* message, the MB must first determine the status of the group in the DS domain. If the group already exists in the domain (has ingress or egress points), the MB does not need to search for the group. However, if the multicast group does not exist in the DS domain, the MB must locate the multicast group. For cases where the join request is using SSM or CBT (Core-Based Trees [93]) where a source address is included, the problem is vastly simplified as the location of the multicast group is easily identified. In other cases, the MB must search for the multicast group using a multicast inter-domain routing protocol such as MBGP [85] or MSDP [84].

The actual edge routers to which the new egress point (ER_{NEW}) can be grafted to is discussed in more detail in Section 4.2. A simple (but not optimal) solution is to graft the new egress point to the closest on-tree edge node (ER_{GRAFT}) that satisfies the PHB precedence of the new egress point. The topology information for such a decision would be readily available from the underlying link state routing protocol (OSPF [80] or IS-IS [81]).

Graft: After the MB has determined the appropriate grafting point; the MB will inform both the new egress point (ER_{NEW}) and the upstream edge router (ER_{GRAFT}) of the change in the multicast tree. If the tree is not being rearranged, only the two edge routers involved, the MB, and the BB (optional) need to know about the change in the multicast tree. The MB sends an *EBM-Tree-Update* message to the two edge routers that contains the ID of the new edge router (leaf), the ID of the parent edge router, the ID of the multicast group, and the PHB for the new branch. Both of the edge routers will update their replication information and future data transmissions to the group will now flow to the new egress point (ER_{NEW}).

In the event that non-ingress edge routers do not keep replication state information (i.e. tree encapsulation), the MB will send the *EBM-Tree-Update* message to the ingress router rather than the grafting point. The advantage of employing tree encapsulation is that tree rearrangement can be conveyed via a single packet to the ingress router(s) rather than all

affected nodes. However, this approach is best suited for highly dynamic groups as there is an additional cost for all data packets due to the overhead of carrying edge replication information. For single sourced multicast trees (such as SSM) and groups with high amounts of group dynamics, such an approach may be advantageous for faster tree optimization (tree rearrangement, etc.). In addition, the use of encapsulation reduces the amount of state information that non-ingress edge routers must store/maintain.

4.1.4 EBM Egress Leave

When an edge router wishes to leave the multicast group, it sends an *EBM-Leave-Request* message to the MB. It is assumed that such a message will only be sent when the edge router has no additional downstream receivers (i.e. the last of its downstream receivers sends a prune or times out). The member leave operation is less critical from the user perspective as the user no longer desires service from the multicast group. However, from the perspective of a service provider, the quick execution of a member leave operation minimizes wasted resources.

Similar to the join operation, the prune operation can be divided into three steps:

- *Inter-domain leave* - The last downstream receiver leaves and the edge node (egress) requests to leave the group.
- *MB Processing* - The MB examines the multicast tree and begins the prune process.
- *Prune* - The MB prunes the appropriate edge routers from the multicast tree.

Inter-domain leave: As with the join request, the leave request arrives at an edge router (ER_{OLD}). Upon receiving a leave request, the edge router should only continue the leave if all of its downstream receivers have left the group. In such a case, the edge router sends an *EBM-Leave-Request* message to the MB. The message includes the edge router ID (ER_{OLD}), the multicast group ID, and the inter-domain leave information.

MB Processing: The processing of the *EBM-Leave-Request* message is vastly simplified compared to the processing of the join request. Whereas the join request may involve inter-domain routing to locate the multicast group, the multicast group already exists in the domain

when the *EBM-Leave-Request* is received. The simplest approach to the leave algorithm is to simply graft any downstream egress points to the upstream edge router of ER_{OLD}. If the ER_{OLD} is a leaf (no downstream egress points), the link is simply pruned.

Prune: Once the MB has determined how to update the multicast tree, it sends an *EBM-Tree-Update* message to the modified upstream point(s) and an acknowledgement to the pruned node (ER_{OLD}). In the event that tree encapsulation is employed, only the ingress router(s) and the pruned router need to be notified.

4.2 Edge-Clustered Trees (ECT)

The construction of multicast trees in an environment such as DiffServ is governed by several constraints that must/should be obeyed. These constraints include:

- *PHB priority:* An egress point with a higher priority PHB must not sit downstream from an egress point with a lower PHB. For example, a packet cannot be tunneled to a BE (Best Effort) egress point with the BE PHB and then tunneled to an EF (Expedited Forwarding [10]) egress point using the EF PHB.

- *PHB promotion:* Conversely, an egress point with a lower priority PHB should not receive a higher priority PHB for its packets than requested. For example, a packet should not be tunneled to an AF (Assured Forwarding [11]) egress point with the EF PHB and then tunneled to an EF egress point with the EF PHB.

- *Minimal hop count:* Packets should be delivered with the minimal hop count possible since no shaping/scheduling is done to balance the cost of additional hops (i.e. shorter path is better than a longer path).

Because of these constraints, traditional multicast tree construction algorithms such as KMB [82] and others cannot be applied to the problem directly. Although it is possible to model the problem as a Steiner tree problem, such a subject is a topic for future research. Since existing algorithms are not sufficient for addressing such constraints, a novel approach called the Edge Clustered Tree (ECT) was developed for constructing trees in such an environment.

4.2.1 ECT Algorithm

The premise of the ECT algorithm is fairly simple, cluster similarly QoS-classed egress points together in an effort to balance the cost of the tree versus the additional hops required for edge-based branching. The ingress node tunnels the packets to clusters that then tunnel the packets to the egress points in their clusters and other downstream clusters. The ECT algorithm itself can be broken into two key phases, cluster construction and cluster linkage.

4.2.2 Cluster Construction

In the first phase, a cluster is constructed centered on each egress point of the multicast group (see Figure 4.2). A cluster consists of all other egress points within H hops of an edge router (E_X) whose PHBs can be satisfied by the PHB used for packets sent to E_X[1]. The cost of the cluster consists of the costs of tunneling to the nodes in the cluster (from E_X to the cluster nodes) and the cost of the tunnel from the ingress router to E_X. The metric $M(E_X)$ for evaluating a cluster centered at node E_X is defined as:

$$M(E_X) = \frac{D(I_G, E_X) + \sum_{i=0}^{|C_X|} D(E_X, C_{X,i})}{|C_X|}$$

where $D(X, Y)$ is the number of hops between X and Y, I_G is the ingress router for the group (single source), $|C_X|$ is the number of nodes within H hops that are satisfied by E_X's PHB, and $C_{X,i}$ is the ith node in the cluster centered around E_X. A lower value of $M(E_X)$ denotes that the average cost of servicing the nodes in a cluster is lower as well. The ingress node is a special cluster/node whose PHB satisfies any egress point.

4.2.3 Cluster Linkage

In the next phase, the clusters are linked together via tunnels to connect the multicast distribution tree for the domain (see Figure 4.3). The algorithm proceeds by connecting the best cluster according to the metric. In the first iteration, only the ingress cluster may be

[1]The precedence of various PHBs is a topic beyond the scope of this paper. We assume that rules to govern such precedence exist in the MB.

```
ConstructCluster (E_Members, E_X, C_X, H)
  i = 0
  for each E_Y in E_Members where E_X != E_Y
    if D(E_X, E_Y) < H AND
     PHB(E_X) ≥ PHB(E_Y) then
       C_X,i = E_Y
       i = i + 1
    end if
  end for
end
```

Figure 4.2 Cluster construction algorithm

tunneled from (although its cluster may not be selected). Once a cluster is selected, the cluster becomes available as a candidate for future clusters to be tunneled from provided that the PHB priority is still satisfied. Ties are resolved by connecting the highest PHB first in order to maximize the chances that other clusters may be tunneled from that cluster.

When a cluster is selected, all of the nodes inside of the cluster are removed from consideration as members of other clusters. The metric of the remaining clusters is recomputed based on the new cluster membership and the potential new grafting points of previously selected clusters (cluster centers only). An additional constraint may be imposed, cluster depth D, such that a cluster may not be more than D tunnels away from the ingress point, thus capping the maximum tunnels to an egress point at $D + 1$ tunnels (tunnels to cluster+tunnel from cluster).

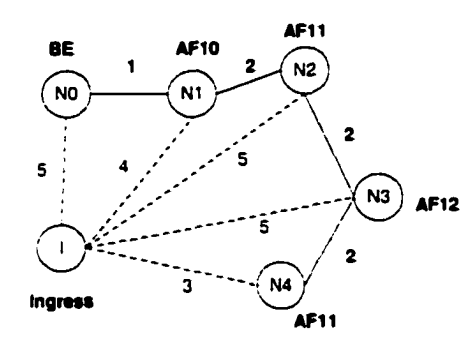
4.2.4 ECT Example

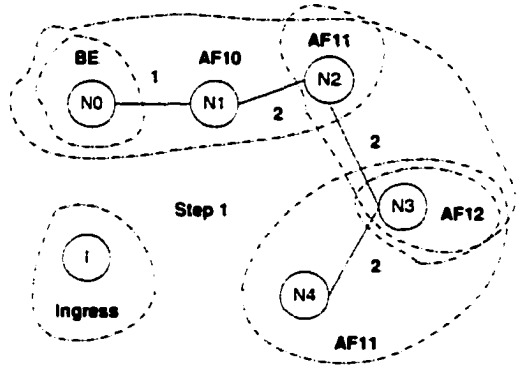
Figure 4.4 shows an example of the ECT algorithm with $H = 2$. In the figure, only paths between edge nodes that are egress points (members) of the group are shown. Each link label denotes the hop count between the nodes.

- *Step 1:* The clusters around each node are constructed considering all egress points when $H = 2$. For the cluster centered on $N1$, it finds two nodes within 2 hops whose PHB is still satisfied by its PHB ($AF10$). Conversely for $N0$, although it can find $N1$ within 2 hops, it cannot offer service ($BE < AF10$). The figure shows a total of 6 clusters.

LinkClusters ($E_{Members}, H, D$)
  $C_{Final} = \emptyset, C_{Unlinked} = \emptyset$
  for each E_X in $E_{Members}$
    ConstructClusters ($E_{Members}, E_X, C_X, H$)
    Add C_X to $C_{Unlinked}$
  end for
  while $C_{Unlinked} \neq \emptyset$
    Compute $M(C_X)$ for all C_X in $C_{Unlinked}$
    C_{Best} = the best C_X out of $C_{Unlinked}$
    Add C_{Best} to C_{Final}
    Connect C_{Best} to upstream node
    Connect members C_{Best}
    Remove all nodes in C_{Best} from all clusters in $C_{Unlinked}$
    Remove all empty clusters from $C_{Unlinked}$
    Check all C_X in $C_{Unlinked}$ for a closer cluster in C_{Final}
     subject to D
  end while
end

Figure 4.3 Cluster linkage algorithm





	Node	Size	Cost	Metric
	Ing	1	0	0.00
	N1	3	1+2+5=7	2.33
Step 2a	N4	2	2+3=5	2.50
	N0	1	5	5.00
	N2	2	2+5=7	3.50
	N3	1	5	5.00

	Node	Size	Cost	Metric
	N1	3	1+2+5=7	2.33
Step 2b	N4	2	2+3=5	2.50
	N0	1	5	5.00
	N2	2	2+5=7	3.50
	N3	1	5	5.00

	Node	Size	Cost	Metric
Step 2c	N4	2	2+3=5	2.50
	N3	1	5	5.00

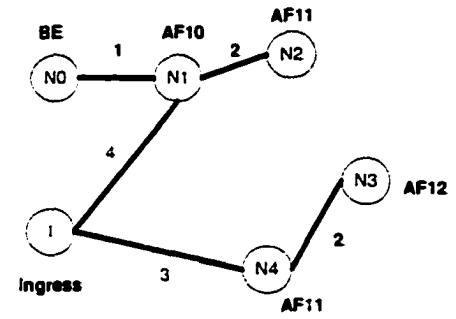


Figure 4.4 Example of Edge Cluster Tree (ECT) construction

- *Step 2a:* The best cluster is linked to the tree. In this case, the ingress point (I) is the best cluster.

- *Step 2b:* The next best cluster is centered on $N1$. The cluster is linked to the tree via the ingress point and the nodes within $N1$'s cluster ($N0$ and $N2$) are removed from all other clusters. The remaining clusters ($N3, N4$) check to see if $N1$ is closer than the ingress point.

- *Step 2c:* The next best cluster is centered on $N4$. The cluster is linked to the tree via the ingress point and node $N3$ is removed from the remaining cluster.

After Step 2c, the tree is constructed covering all the egress points for the multicast group and satisfying the PHB priority rules. The net cost of the ECT is 12, significantly lower than the other stateless option of ingress-only unicasting at a cost of 22. The tree constructed by the simple EBM algorithm (mentioned earlier) would depend upon the join order of the egress points.

4.2.5 Cost of EBM

The running time of the ECT algorithm is fairly easy to calculate. For the cluster construction phase, the running time is directly dependent upon the number of edge nodes in the network. With N edge nodes present, the algorithm will take $O(N^2)$ time to complete. At worst case, all other nodes can be found within H hops, thus requiring $N-1$ PHB comparisons for cluster construction. Since N clusters are originally constructed, the cluster construction phase will take $N \times N$ or $O(N^2)$ time.

For the cluster linkage phase, the worst case input occurs when N clusters are given to the algorithm. In this case, none of the nodes can be sub-clusters (tunneled from) of other clusters. Thus, N steps must be applied to link all of the clusters with each new step scanning all previously added clusters (worst case N at the last cluster) and also notifying all other clusters of the members contained in the cluster (worst case $N-1$ at the first cluster). Thus,

the algorithm will be run N times with N possible comparisons/computations per step resulting in a $O(N^2)$ worst case run time.

4.3 Analyzing the Benefits of EBM

4.3.1 Comparing MBone and EBM

The closest work to the EBM architecture is the original MBone architecture. The EBM architecture differs from MBone in several key respects. First, this solution is specifically geared towards the DiffServ architecture and towards providing a transport functionality only. Whereas MBone is interested in end-to-end service, EBM focuses solely on the unique aspects of transport across the DS domain. Second, due to the close coupling of EBM with DiffServ, EBM allows for unique functionality that cannot be offered with MBone. For example, rather than relying on other ISPs as would occur with MBone, a single ISP using EBM can force authentication for all multicast routing updates in the domain and manage the QoS impacts via the MB. Although EBM at its lowest level can essentially be thought of a specialized topology of MBone, it is the coupling of the MB, the uniqueness of the DiffServ topology, and the ECT algorithm that offer EBM its true distinction versus MBone.

4.3.2 Multicast Deployment, Performance, Security, and Management

Although multicasting support has been deployed in many routers in the global Internet, to date only a select group of routers are actually can be used for multicasting. The EBM approach can help expedite the deployment and adoption of multicasting in the general Internet due to several factors. First, such a scheme is highly appealing to a service provider and also for multicast deployment in general. From the perspective of the service provider, the protocol independence allows the provider to offer multicast without depending upon one's neighbors for end-to-end deployment or tunneling. Consequently, such an approach benefits multicast in general as EBM provides the critical puzzle piece that will interoperate between different multicast protocols. As long as appropriate translation is done at the MB or edge router, multiple multicast protocols can operate together without even knowing the other protocol

exists. In addition, EBM relies on existing IP routing protocols and requires only minimal changes to edge routers. EBM migrates most of the complexities of multicasting to the MB, thus simplifying multicast deployment even further.

Second, the EBM architecture overcomes the standard performance penalties of traditional IP multicasting. Since multicast packets are treated exactly the same as unicast packets and incur zero additional processing or state overhead, core routers can remain entirely multicast-unaware. Thus, the classification and routing functionality of the core routers do not need to change, a potentially significant savings in cost since the complexity of multicast is eliminated. In addition to the reduced processor load, the complete elimination of multicast from the core increases the scalability of the core routers dramatically.

Third, security can be inherently included in the EBM architecture from the beginning. The transport approach of EBM allows the security to be applied on a domain-wise basis rather than an end-to-end basis, thus dramatically reducing the scale of the problem. Without modification, the multicast-everywhere approach of traditional IP multicasting has the potential to be the ultimate in Denial of Service (DoS) attacks due to the fact that no policing occurs in traditional IP multicasting. However, in the EBM architecture, the inputs to the multicast group can be policed (at the edge routers) and the multicast operations themselves (graft, prune) can be securely protected. Since core routers are multicast unaware, only unicast packets could be sent and thus, no additional replication could occur. In the worst case, a malicious attacker could compromise the edge router itself, thus being free to replicate packets as necessary. However, in such a case, the entire integrity of the DiffServ network would be compromised, a much more serious problem than simply multicasting.

Fourth, the EBM architecture lends itself well to multicast resource and traffic management. In addition to being able to police and manage traffic at the edge of the network (prevent malicious or non-conforming traffic), the entire impact of a multicast tree is known and available at the MB. As a result, EBM is able to efficiently manage the impact of multicasting on the network. For any network that must provide a wide range of QoS, such management is essential.

4.4 Simulation Studies

In order to evaluate the performance of the EBM architecture, several extensions to the ns-2 simulator [90] for EBM, DSMCast, ingress-branching, and PIM-SM [68] were developed. These modules were developed using the GenMCast (Generalized MultiCast) extensions for ns-2 [91]. For the simulations, the performance of the following models were evaluated:

- *Ingress-branching* - Packets are tunneled out from the ingress node to all egress points in the DS domain. Replication is done only at the ingress node.
- *EBM* - The EBM model is studied using two variations, the simple algorithm (outlined in Section 4.1) and ECT.
- *DSMCast* - The adaptive DSMCast model [74] is studied that selects either DSMCast or ingress-only branching depending upon which is the least cost approach.
- *PIM-SM* - Although the state-based approach of traditional IP multicasting is not scalable, the PIM-SM protocol provides an excellent baseline for evaluating the relative overhead of the algorithms versus the best possible case for performance.

The simulations were conducted for a single domain scenario of varying random network topologies and varying QoS distributions (uniform and non-uniform). The models were evaluated on three performance metrics, the average bandwidth consumed per link, the average number of hops, and the average number of tunnels to an egress point. The average bandwidth consumed per link gives an indication of the relative impact of multicast traffic on each link provided that no multicast packets are dropped. The average hops and average tunnels metrics demonstrate the additional impact of tunneling that occurs since packets may take a longer route to reach edge nodes versus ingress-only tunneling. The parameters for the simulations were as follows:

- The network topology consisted of 32 core nodes connected with an average node degree of 3. 16 edge routers connected by at least one core router in between two edge routers. NSFNet was also used as a realistic topology for comparison [92].

Table 4.1 EBM ns parameter list

Parameter	**Value(s)**
Link Bandwidth/Delay	1.0 Gb/s, 5 ms
PHBs	AFxy (x=1..3,y=1..4)
Uniform	x(25%,25%,25%,25%),y(33%,33%,33%)
Non-Uniform	x(10%,10%,40%,40%),y(10%,30%,60%)

- Each edge router was the source for two SSM multicast groups (32 total groups).
- Each multicast group produced CBR traffic with an exponentially distributed packet size and packet inter-arrival time. Unless otherwise noted, the mean for the values were 400 bytes and 100 ms.
- The starting number of egress points for each simulation was 160 egress points (average group density of $\frac{160}{32} = 5$ egress points) with the egress points randomly distributed between the various groups.
- The events that constitute group dynamics (join/leave) were uniformly distributed with a mean inter-event time ranging from 150 ms to 250 ms. The probability for a join or leave was 0.5 for each.
- Packets in EBM use the state-based approach whereby per-group state information for replication is kept at edge routers.
- The settings for ECT were $H = 3$ and $D = 2$.
- The network was assumed to have sufficient buffer space and bandwidth such that packet dropping due to buffer overflows does not occur. This is done to ensure an adequate comparison of the overhead of EBM.
- Additional parameters are summarized in Table 4.1.

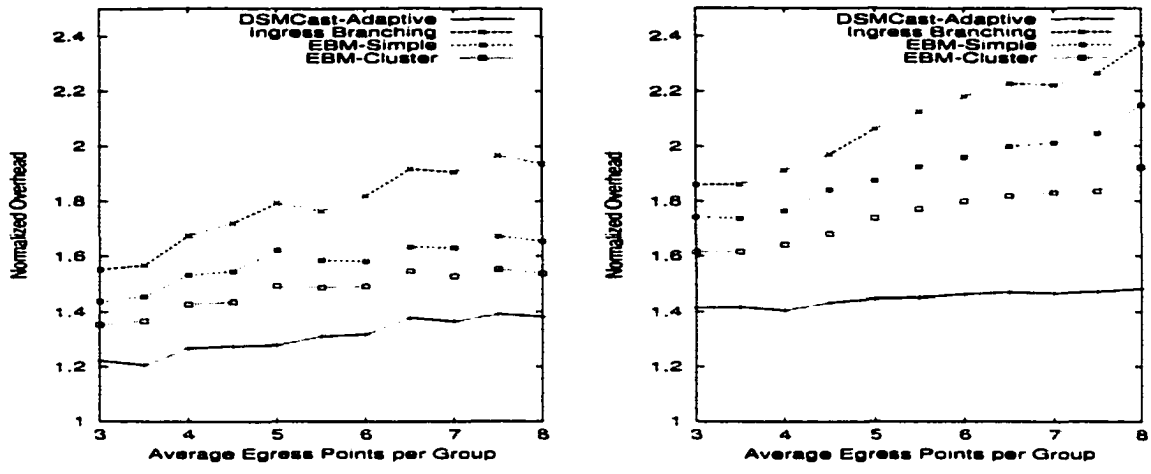
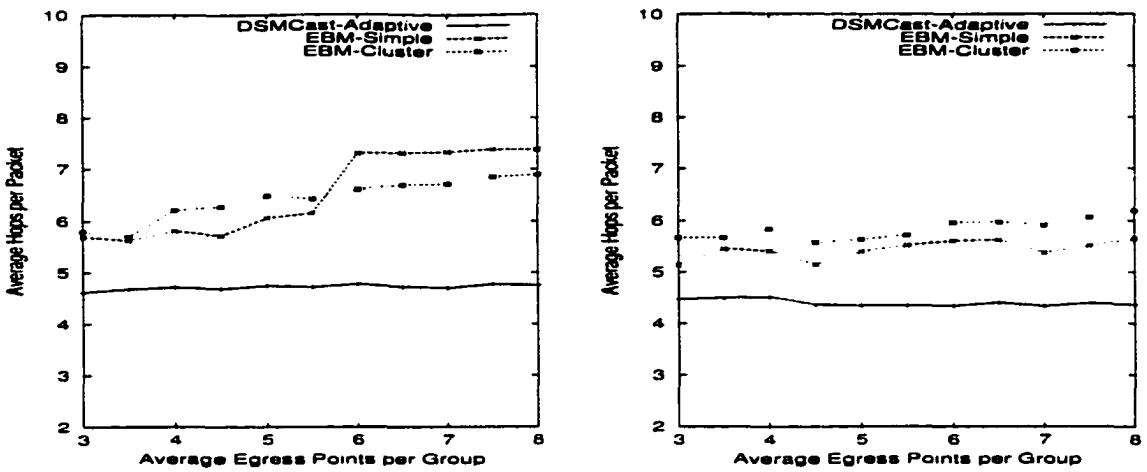


Figure 4.5 Effect of group size on overhead - uniform distribution - normalized - (a) Random network (b) NSFNet

4.4.1 Effect of Group Density

In Figure 4.5, the impact of the group size (egress points) on the performance of the various approaches is shown. In order to better study the additional overhead of core statelessness, the results are as a ratio to the cost of traditional IP multicasting (PIM-SM). DSMCast follows traditional IP multicasting the closest as it uses an actual multicast tree embedded inside the packet, thus achieving much of the benefit of the multicast tree. Next, the two EBM approaches follow with ECT offering a marginal improvement in both cases. As will be seen later, the difference in performance is due to the fact that the performance parameters of ECT are properly tuned.

Figures 4.6 and 4.7 examine the effects of group size on EBM in more detail. Figure 4.6 offers additional insight as to why EBM performs worse than DSMCast. Whereas DSMCast manages to keep its hop count fairly constant in both the random topology and NSFNet with increasing egress points, EBM suffers from an increase of average hop count. Due to the fact that EBM does not replicate in the core, each additional egress point will increase the average hop count due to the fact that the hop count to reach an individual node is cumulative from



(a) (b)

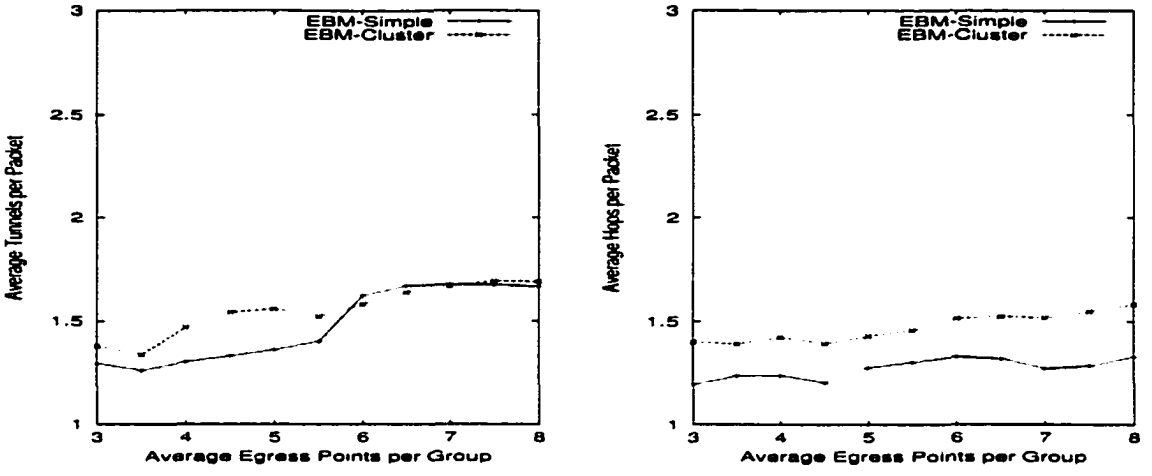
Figure 4.6 Effect of group size on average hop count - uniform distribution - normalized - (a) Random network (b) NSFNet

the previous tunnels rather than being directly sent from the ingress. The same observations can be gathered from Figure 4.7 where an increase the average tunnel count is correlated with an increase in the average hop count.

For the case of ECT, an increase in hop count implies that ECT is functioning better. When the hop count is increased, it implies that ECT is appropriately locating clusters and tunneling from those clusters. However, when ECT is not tunneling from clusters, its performance degrades to closer to that of ingress-branching, thus offering a performance worse than that of the simple algorithm. The findings in Figure 4.8 only accentuate the performance metrics of the two EBM models. When ECT is operating properly, the clusters offer a better cost than simply joining the closest node as with the simple case.

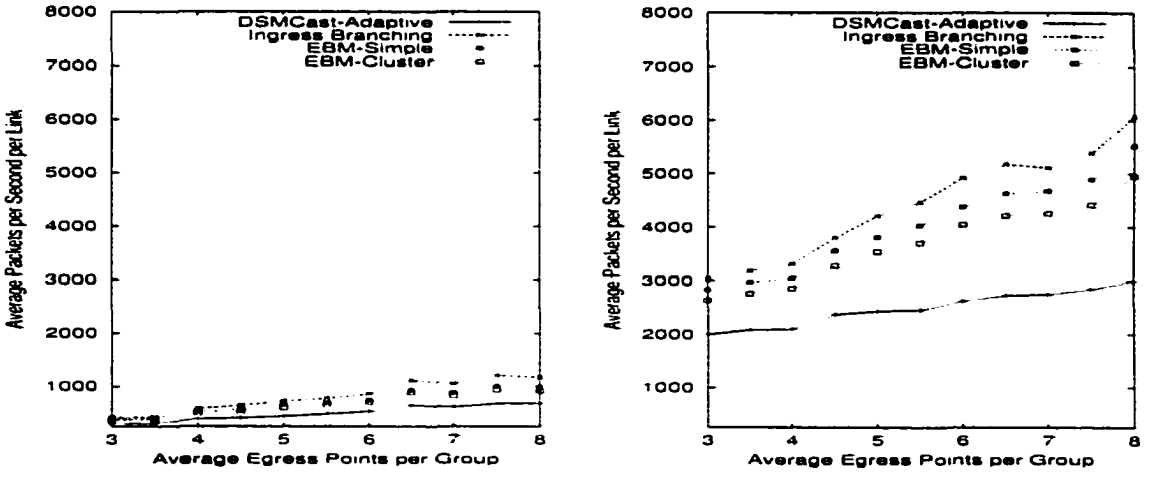
4.4.2 Effect of Packet Size

Figure 4.9 examines the effect of average packet size on the performance of EBM. Unlike DSMCast whose overhead decreases significantly with increased packet size, the decrease in overhead of EBM follows the ingress-branching approach much more closely. Since EBM does not allow for core-based replication, the increase in packet size only emphasizes the additional



(a) (b)

Figure 4.7 Effect of group size on average tunnel count - uniform distribution - normalized - (a) Random network (b) NSFNet



(a) (b)

Figure 4.8 Effect of group size on transmitted packets - uniform distribution - normalized - (a) Random network (b) NSFNet

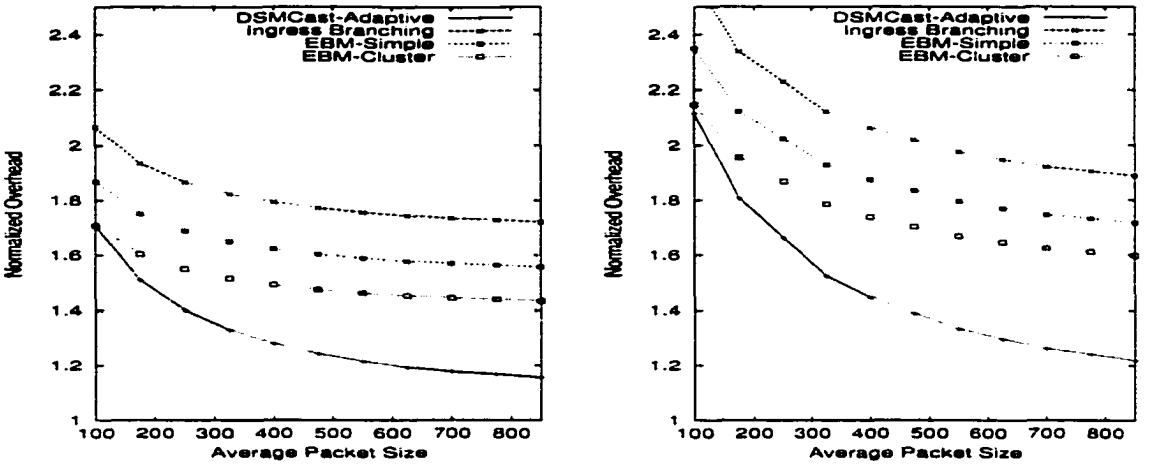


Figure 4.9 Effect of packet size on overhead - uniform distribution - normalized - (a) Random network (b) NSFNet

hops that packets must take on the network. In addition, the increase in packet size further differentiates the performance differences between the simple join algorithm of EBM and EBM using ECT.

4.4.3 Effect of Group Size - Non-Uniform PHBs

Figure 4.10 plots the performance of the various models as a non-uniform traffic model is applied while the group size is varied. Although the effect is not noticeable on the graphs, the change of distribution of egress PHBs has a small impact on the performance of EBM (less than 1%). As a result, the effect of the various ECT parameters is only analyzed in the presence of a uniform egress PHB distribution.

4.4.4 Effect of ECT Parameters

4.4.4.1 Effect of H (hops) setting

Figure 4.11 examines the impact of the H (hops) setting on the performance of the ECT algorithm with the performance of DSMCast, the simple EBM algorithm, and ingress-only

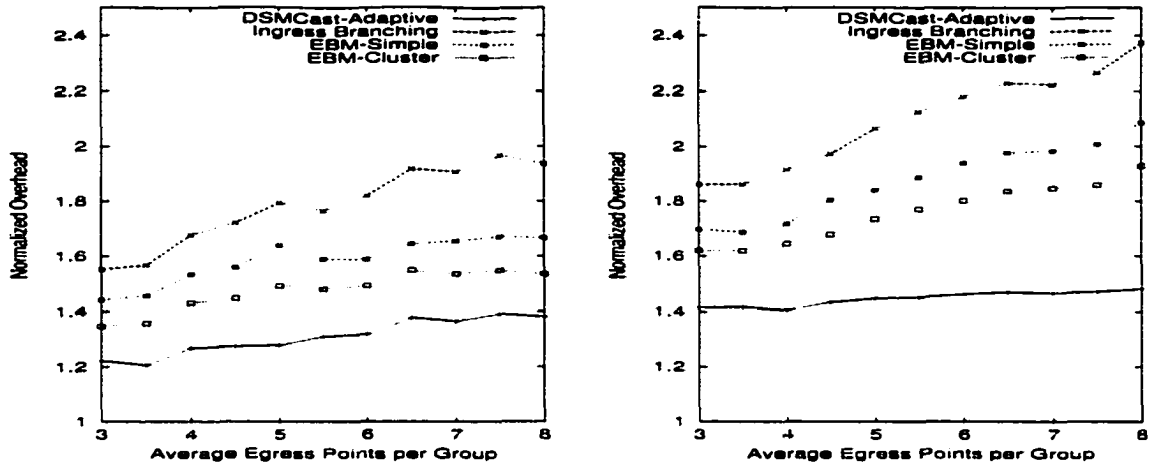


Figure 4.10 Effect of group size on overhead - non-uniform distribution - normalized - (a) Random network (b) NSFNet

branching included as a baseline. As the number of hops increases, we notice a decrease in the cost of the trees constructed by ECT. When $H = 0$ or $H = 1$, the algorithm performs similar to the simple EBM algorithm since only selected clusters can allow intermediate tunneling. With an increase in H, the cluster algorithm is able to find more nodes within reach and hence reduce the overall cost of the distribution tree. However, beyond a certain point ($H = 3$), the ECT algorithm begins to perform worse. At this point, the clusters begin to contain too many nodes and nodes are absorbed that would be better optimized by being in separate clusters. Once H passes the maximum width of the random network $H = 6$, nearly all nodes are within range of one another.

The effect of the ECT hops setting goes beyond the cost of the multicast tree. Figure 4.12 shows the tunneling impact on groups of AF (Assured Forwarding) classes by the hops setting (AF1x = AF10, AF11, AF12). At the point where the average number of tunnels for all classes is maximized ($H = 3$), the best performance (see Figure 4.11) is also achieved as well. Once the threshold of $H = 3$ is passed, the clusters contain too many nodes as is evidenced by the lower average tunneling value. In fact, even the lower classed AF PHBs dramatically lower their average tunneling value and begin to converge together. In actuality, it is when the

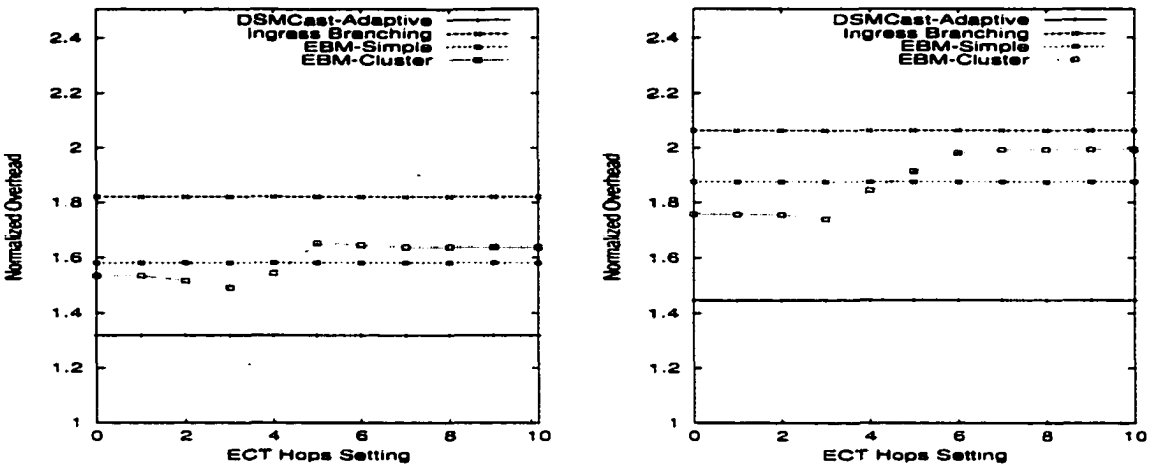


Figure 4.11 Effect of cluster hops on overhead - uniform distribution - normalized - (a) Random network (b) NSFNet

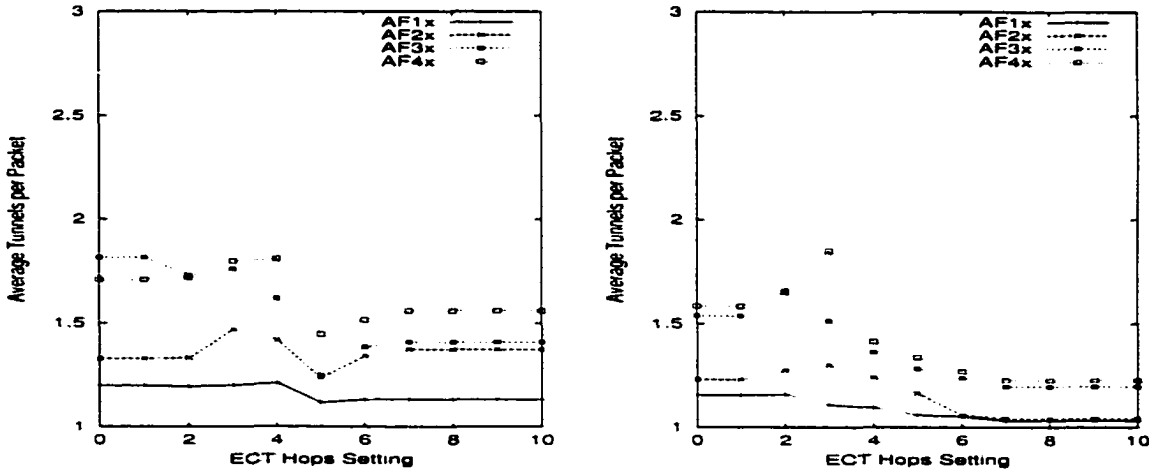


Figure 4.12 Effect of cluster hops on class-wise tunnel count - uniform distribution - normalized - (a) Random network (b) NSFNet

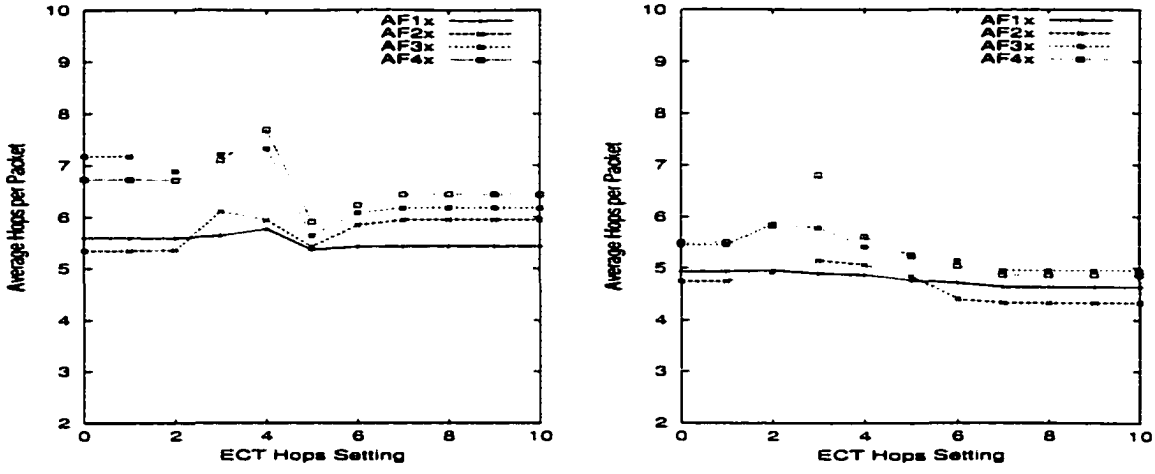


Figure 4.13 Effect of cluster hops on class-wise hop count - uniform distribution - normalized - (a) Random network (b) NSFNet

opposite effect (increased average tunneling between nodes) is seen, a more efficient multicast distribution tree is achieved. The average hop count for the various classes is shown in Figure 4.13. Although the graph is not shown, a similar trend holds as well for the non-uniform PHB distribution. In addition, the effect of the depth parameter (D) is not shown is it has only a negligible impact on EBM.

4.5 Conclusions

In this chapter, a new architecture for DiffServ multicasting, Edge Based Multicasting (EBM), was proposed that relies on edge-based replication, tunneling, and a Multicast Broker (MB) entity to deliver a scalable multicast transport service across a single DiffServ domain. In addition, a novel algorithm, the Edge Cluster Tree (ECT), was proposed that captures the unique aspects of heterogeneous QoS management in a DiffServ domain.

The EBM architecture has tremendous potential for expediting the wide scale deployment of multicasting as DiffServ is deployed. The architecture minimizes the impact of multicasting support by requiring only modifications to the edge routers and the inclusion of a Multicast

Broker. For source-specific groups, the MB can simply be included as part of the ingress router. Thus, the EBM architecture represents a viable approach for joining the two technologies of DiffServ and multicasting.

CHAPTER 5. QOS HETEROGENEITY

While the previous two chapters proposed new architectures for DiffServ multicasting and focused on the overhead/practicality of the two architectures, this chapter addresses the effect of the various approaches on the user QoS. Specifically, the issue of QoS is complicated by the need for support of heterogeneous QoS in the multicast tree. This chapter investigates the effect of sender-driven QoS and the intricacies of single tree support for heterogeneous QoS. In addition, this chapter provides the first in-depth simulation studies of the effect of single tree heterogeneous QoS support.

This chapter is organized as follows. Section 5.1 describes the heterogeneous QoS problem in more detail and presents several applications and examples that use heterogeneous QoS. Next, Section 5.2 discusses several solutions and their approaches to the heterogeneous QoS problem. Then, Section 5.3 investigates the implications of heterogeneous QoS support through detailed simulation studies. Finally, Section 5.4 offers several concluding remarks.

5.1 Problem

Due to the fact that different multicast receivers may require different levels of QoS, it is only natural to offer support for heterogeneous QoS via the DiffServ (DS) domain. When providing heterogeneous QoS for multicasting, the underlying approach for multicasting directly affects the quality/complexity for resolving heterogeneous QoS. Whereas multicast uses an inherently receiver-driven QoS, DiffServ provides a sender-driven QoS. The sender-driven QoS arises from how the QoS for packets is selected in a DS domain. In contrast, multicasting pushes QoS from the receiver to the sender, allowing the routers in the path to adapt the QoS, rather than forcing the sender to be aware of the heterogeneous QoS requirements of the

receivers. This abstraction allows the sender to operate independently of the heterogeneity of the receivers, thus allowing the sender (source) to service millions of heterogeneous receivers without being of aware of the QoS requirements of each individual user. Unless the sender becomes receiver-aware (a non-scalable option), a mechanism must be developed to bridge the gap between the sender-driven QoS of DiffServ and the heterogeneity of the downstream receivers.

5.1.1 Heterogeneous QoS - An Example

Although heterogeneous QoS support was included from the beginning in IntServ and RSVP, the issue of heterogeneous QoS in DiffServ multicasting has only been considered sparingly [40, 67, 41]. Thus, one of the first tasks is to define the applications that would take advantage of such network services and from these applications the problem that must be addressed.

5.1.1.1 Benefits of Heterogeneous QoS

The support for heterogeneous QoS in multicasting can have significant implications in multicast application design. First, in-tree heterogeneous QoS offers the benefit of only having to manage a single group at the sender. The in-tree heterogeneous QoS offers additional flexibility for differentiation that does not have to be explicitly designed into the application. Rather than being limited by the options offered by the application (i.e. different compression rates, etc.), support for in-tree heterogeneous QoS allows the network to adapt to the user rather than the application adapting to the user.

Second, the support for heterogeneous QoS can have significant implications on bandwidth utilization. Since most multicasting content tend to be geared towards high bandwidth or rich multimedia streams, the cost of offering different groups for different QoS levels can be significant. Although multicasting reduces the overall load on the network quite significantly, the use of multiple QoS levels has a profound effect on the last hop before the datacenter, typically a bottleneck for network performance in the first place.

5.1.1.2 Application Space

Consider the following examples: an audio streaming event and a server farm for a large multiplayer on-line game. Suppose in the first example that the provider is streaming the audio broadcast of a football game with 10,000 users listening to the game throughout the Internet. The service provider wishes to offer different levels of service: gold, silver, bronze, and free service. The first option is simply to use different codecs and compression rates to offer the service. However with heterogeneous QoS, the same service can be offered using a single codec. Rather than streaming out via different groups, the heterogeneous QoS is supported in a single tree and the PHB of the packet changes as it traverses the multicast tree. If the service provider is only offering a single event, it may an acceptable tradeoff to simply use multiple codecs.

However, consider the case of a large multiplayer on-line game whereby a server farm is hosting multiple on-line games, each employing an individual multicast group of up to 64 or 128 users. Suppose that the game service offers 10 different tiers of service to maximize game performance for different components of the game. In such a case, all of the players are listening to the same group but require different QoS levels depending upon where the player is at in the game. Unlike the first case where only a single group was being managed, the server farm may be hosting thousands or even tens of thousands of games. As the number of supported groups increases, the desirability of in-tree heterogeneous QoS increases as well since it can offer significant performance improvements.

5.1.1.3 Example Network using Heterogeneous QoS

Consider the example network listed in Figure 5.1. In this network, each receiver is requesting distinctly different service levels. Whereas R_1 is requesting AF10 (Assured Forwarding [11]) level service. R_2 is requesting only BE (Best Effort) level service. Since both receivers share the same path from the ingress node, the difference must be resolved. The same differentiation applies for both R_3 and R_4 and for all receivers sharing the first hop from the ingress router.

In the DiffServ architecture, packets are marked at the ingress router or sender and the

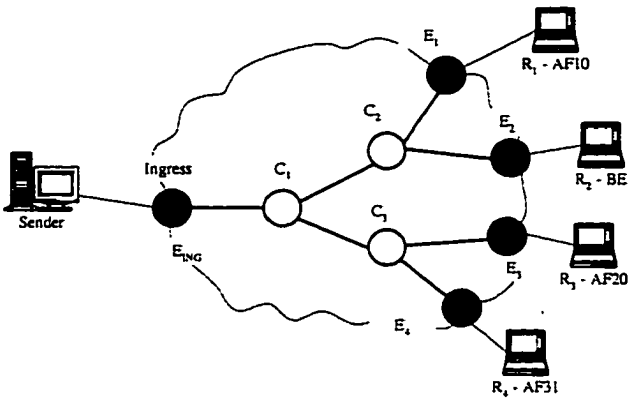


Figure 5.1 Example of heterogeneous QoS in a DiffServ domain

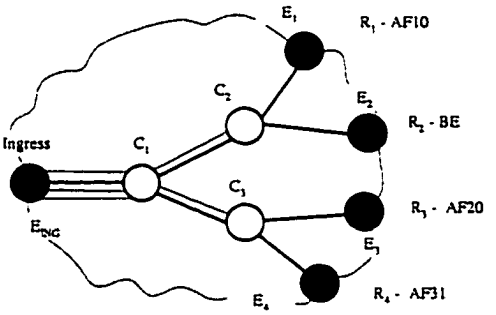


Figure 5.2 Example of heterogeneous QoS in a DiffServ domain - separate trees

PHB (denoted by the DSCP) does not change as the packet traverses the DS domain. Since the marking does not change once the packet leaves the ingress router, DiffServ provides an absolute sender-driven QoS rather than a flexible receiver-driven QoS (such as IntServ). However, as shown in the figure, the absolute sender-driven approach of DiffServ has problems with heterogeneous QoS requirements for receivers.

Without the option for dynamic PHBs in the DiffServ domain, the simple solution would be to send separate packets for each unique QoS level (PHB) in the group. Depending upon the relative heterogeneity of members and the location of such members within groups in the DS domain, this solution may waste excessive amounts of network bandwidth. As shown in

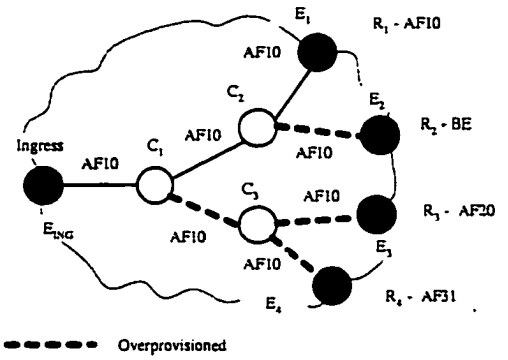


Figure 5.3 Example of heterogeneous QoS in a DiffServ domain - over-provisioning

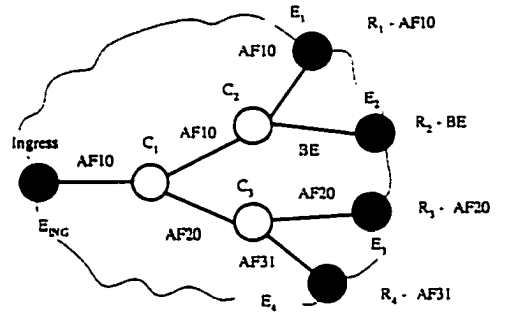


Figure 5.4 Example of heterogeneous QoS in a DiffServ domain - dynamic PHB

Figure 5.2, the multicast service would downgrade to the equivalent of separate unicasts.

In contrast, a second solution is to require all receivers for the group (egress nodes) in the domain to use the highest DSCP for the group (see Figure 5.3). This solution suffers from two problems in that the solution unnecessarily consumes resources and that all of the receivers in the group may not be willing to pay for the highest level of QoS.

From the perspective of the service provider, the only cost of adding R_2 should be the last link to R_2, provided that R_1 is willing to pay the full cost for AF10 service from E_{ING} to R_1. As long as all upstream links from a receiver meet the necessary QoS of the new receivers, the cost of adding additional receivers for the group which share those same links is zero for the

shared links (from E_{ING} to C_2). The cost is zero because the service is being sent regardless of whether R_2 is present or not. Thus, the optimal solution is to introduce a scheme for multicast whereby the DSCP (and hence the PHB) is allowed to change in the network and adapt to the needs of the downstream receiver. Figure 5.4 shows the same example with dynamic PHBs allowed.

5.1.2 Prioritizing PHBs

One of the difficult problems associated with heterogeneous QoS is how to prioritize various PHBs. The main problem is that PHBs may perform different depending upon the network load. For instance, consider several of the following examples:

- *EF vs. AF*: Whereas EF (Expedited Forwarding) provides a low loss/low delay PHB that is superior to AF under a significant network load, EF does not always provide superior performance. Since EF is rate-limited, it may actually offer worse performance than AF on a lightly loaded network.
- *AF11 vs. AF20*: Whereas the AF11 class offers a better scheduling delay than AF20, the AF11 has a worse drop precedence than AF20. Depending upon the relative load of AF20, the loss rate of AF11 may actually penalize the AF20 class.

The combination of PHBs present interesting dilemmas that can either only be resolved by the network administrator. For the EF vs. AF prioritization, the simplest solution may be to not allow the two trees to be combined due to the rate-limitation and low loss/low delay properties of EF. In contrast, a network administrator may feel that AF11 traffic should rule over AF20 traffic and can configure the prioritization mechanisms appropriately. In fact, the result of prioritization may be to use an entirely different PHB (AF10) rather than to use either of the two considered PHBs.

In the end, the decision for prioritization is directly tied to the PHBs in the underlying network. Since the DiffServ working group has only defined the explicit behaviors of EF and BE[1], it is extremely difficult if not impossible to define absolute rules for prioritization. Rather,

[1]Although the loss behavior of AF is defined, the remainder of AF is left to the network administrator.

this chapter simply assumes that prioritization is possible and offers a desirable benefit of the network.

5.1.3 Good Neighbor Effect

The use of dynamic PHBs also introduces the possibility for 'undesired' behaviors. Consider the case of the previous example of R_1 and R_2. Since R_1 and R_2 are separated only by a single link in the multicast tree, the potential exists for the *Good Neighbor Effect*. R_2 benefits from the premium service of its neighbor (AF10) and receives near-$AF10$ like services while paying for only best-effort services.

Although the QoS levels of both R_1 and R_2 are satisfied, the scenario creates a dilemma for a network provider. On the one side, the cost of adding R_2 is negligible (i.e. only the cost of the last link) since R_1 was already paying for the edge-to-edge service. On the other side, the differentiation of the respective network is reduced. Since R_2 would be receiving much closer to $AF10$ service rather than BE service, it is highly unlikely R_2 would want to pay for a higher level service unless the last link becomes congested. Thus, the single multicast tree is open to a *collusion* attack whereby multiple downstream receivers work together to pay the minimal amount for the best QoS.

The simple solution is to use separate trees for different PHBs to deliver the appropriate QoS. However, such a solution is inefficient since the bandwidth savings of a single tree would not be realized. Thus, a network provider is faced with a dilemma, consume less bandwidth and offer potentially unfair services or consume additional bandwidth and offer fair services. Unless the service provider performs negative shaping to balance out the Good Neighbor Effect (discussed more in Chapter 7), the service provider must balance which aspect of the service is more costly.

5.2 Heterogeneous QoS Solutions

In addition to the architectures proposed in this dissertation, several other architectures have addressed the issue of heterogeneous QoS in a multicast tree. For almost all of the works

besides QUASIMODO [67], the approaches follow the initial state-based work of the IETF. Although QUASIMODO does begin to address heterogeneous QoS, the discussions regarding heterogeneous QoS were not discussed in more detail due to space restrictions.

5.2.1 Extending Traditional IP Multicast

In [40], the authors presented a scheme extending traditional IP multicast to support heterogeneous QoS in a DS domain. In order to support heterogeneous QoS, each multicast routing entry would also include a DSCP detailing the codepoint to be used when replicating traffic for the given group onto the specified link. The work specifically outlined support for two levels of QoS, the current group QoS and a LE (Limited Effort) PHB [37] for those that do not wish to use the codepoint of the group.

However, this work suffered from several notable problems. First, the issue of upstream PHB conflicts for full heterogeneous support[2] was left as a management issue. The underlying architecture of traditional IP multicast (i.e. per-group information embedded in core routers) makes the issue of upstream PHB resolution non-trivial. Unlike DSMCast and EBM where the entire tree is known for PHB resolution, PHB resolution must take place at each router along the tree.

For example during a multicast join, one must consider not only the issue of when PHBs should be propagated but also necessary reservation/communication with the BB as the PHB propagates up the tree. In addition, a past history must be maintained of downstream links in order to appropriately degrade the service if the higher classed node leaves the QoS group.

In addition, consider the earlier example network from Figure 5.1. Consider the case where R_3 joins the group first. In this case, the tree has a codepoint of AF20 from edge to edge. Now, if R_2 and R_1 join in succession, a priority problem occurs. If the core nodes along the tree do not propagate the 'better PHB' QoS up the tree, the downstream nodes will not be receiving the QoS that they requested. For the case of R_1 which is requesting AF10 service, it would actually receive QoS of AF20-BE-AF10 as the packets go across the multicast tree

[2]For example, a receiver of a higher priority class joins after a lower class.

if the priority is not propagated upstream. Although the order may not always be the worst case, this simple example shows the necessity of propagating the *best* PHBs upstream.

The propagation of higher priority PHBs introduces the second issue with traditional IP multicasting with PHBs. For each case where the PHB must be propagated, the distributed nature of knowledge of the multicast tree requires that the core router must communicate with the BB. In addition to violating the DiffServ principle of simple core routers, such a scheme also creates the potential for fairly inefficient resource allocation due to excessive message passing.

Consider a case containing N routers from edge to edge of a multicast tree. If the PHB must be changed to a higher value, there is potential for N messages to be sent between the various nodes and the BB of which $N-2$ would originate from a core router. In addition, consider the case where one of the upstream reservations fails. Unless a scheme exists to free such resources downstream in the event of an upstream resource reservation failure, resources would unnecessarily be allocated until freed by the underlying soft state mechanism.

5.2.2 DSMCast and Heterogeneous QoS

In contrast, the use of dynamic PHBs with DSMCast does not suffer from similar problems. As the core routers are kept stateless, propagation of 'better PHBs' is automatically taken care of at the ingress router. This prioritization can be done with full knowledge of the multicast tree. In addition, such information can be easily conveyed to the egress router in the event that egress shaping is also desired. As a result of the centralized location for a given group, only a single message is required from the ingress router to the BB. For cases where the traffic allocation may fail (insufficient resources), the edge-based approach prevents unnecessary resource allocations. Most importantly, the scheme removes the complexity of such operations from the core, thus maintaining the simplicity outlined by DiffServ architecture.

In addition to multicast groups, DSMCast can also be adapted for use with select unicast connections. The notion of a dynamic PHB for unicast packets is a promising but unexplored research area. Whereas DiffServ employs a coarse-grained QoS (i.e. the PHB does not change as the packet traverses the domain), DSMCast could allow unicast packets to use dynamic

PHBs as they cross the domain.

5.2.3 EBM and Heterogeneous QoS

Similar to DSMCast, EBM also addresses heterogeneous inherently in its underlying architecture. The ECT algorithm manages all heterogeneous QoS prioritization and elegantly addresses the need for separate paths when PHBs cannot be combined. The tree construction algorithm is only possible due to the fact that the group tree is computed for the domain from a centralized location (the ingress or the MB), thus allowing the algorithm to fully capture and exploit the heterogeneity of the tree.

Unlike DSMCast, EBM can potentially offer slightly better differentiation between different classes. Since replication and PHB changes only occur at edge routers, there is a better chance with EBM that a packet will need to traverse additional links and hence will be exposed to more links on the new PHB. Conversely, these additional hops may have a negative impact versus DSMCast as the additional hops offer the potential both additional delay as well as additional loss. In addition, since core routers are multicast unaware, heterogeneous QoS for unicast connections would not be possible.

5.3 Simulation Studies

In order to evaluate the effects of single tree heterogeneous QoS, extensive simulation studies were conducted using the *ns-2* [90] simulator and the GenMCast extension for multicasting [91]. The simulation studies compared the performance of four models for heterogeneous QoS. a state-based approach and two approaches using DSMCast. The EBM architecture was not studied due to the fact that EBM packets can take a significantly different path than the other approaches. The differences between DSMCast and EBM are discussed in more detailed in the performance studies of Chapter 4. The models are summarized in more detail below:

- *PIM-SM:* A PIM-SM [68] model was used that employed dynamic PHBs without upstream propagation. This model is used to study the effects of non-propagation[3].

[3] Although non-propagation would not be employed in practice, the effects of non-propagation offer interesting

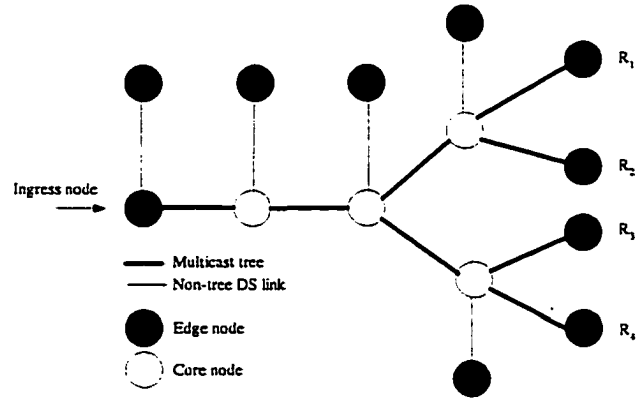


Figure 5.5 Simulation network topology

- *DSMCast - Single Tree:* The first DSMCast model employs a single tree with dynamic PHBs.
- *DSMCast - Separate Trees:* The second DSMCast model employs separate trees for each QoS level (unique PHB).

In order to isolate the performance implications of each model, the network topology in Figure 5.5 was used with two traffic scenarios. A single multicast group was monitored with 4 different receivers, each with a different DSCP of AF10, AF20, AF30, and AF40, respectively. For each of the edge nodes in the DiffServ domain, background traffic using both TCP and UDP was sent to other edge nodes.

The primary performance metric used to evaluate the models was the per-hop delay. The per-hop delay was calculated by dividing the total delay from edge-to-edge by the number of hops that the packet traveled. The simulation studies were also conducted for both uniform as well as non-uniform class distributions for the background traffic as listed in Table 5.1.

5.3.1 Effect of PHB Propagation (Worst Case)

In Figure 5.6, the effect of link bandwidth on per-hop delay is shown for the worst case group dynamics. In this case, R_1 joins first with a PHB of AF40 followed by R_3 with a PHB

insights into heterogeneous QoS.

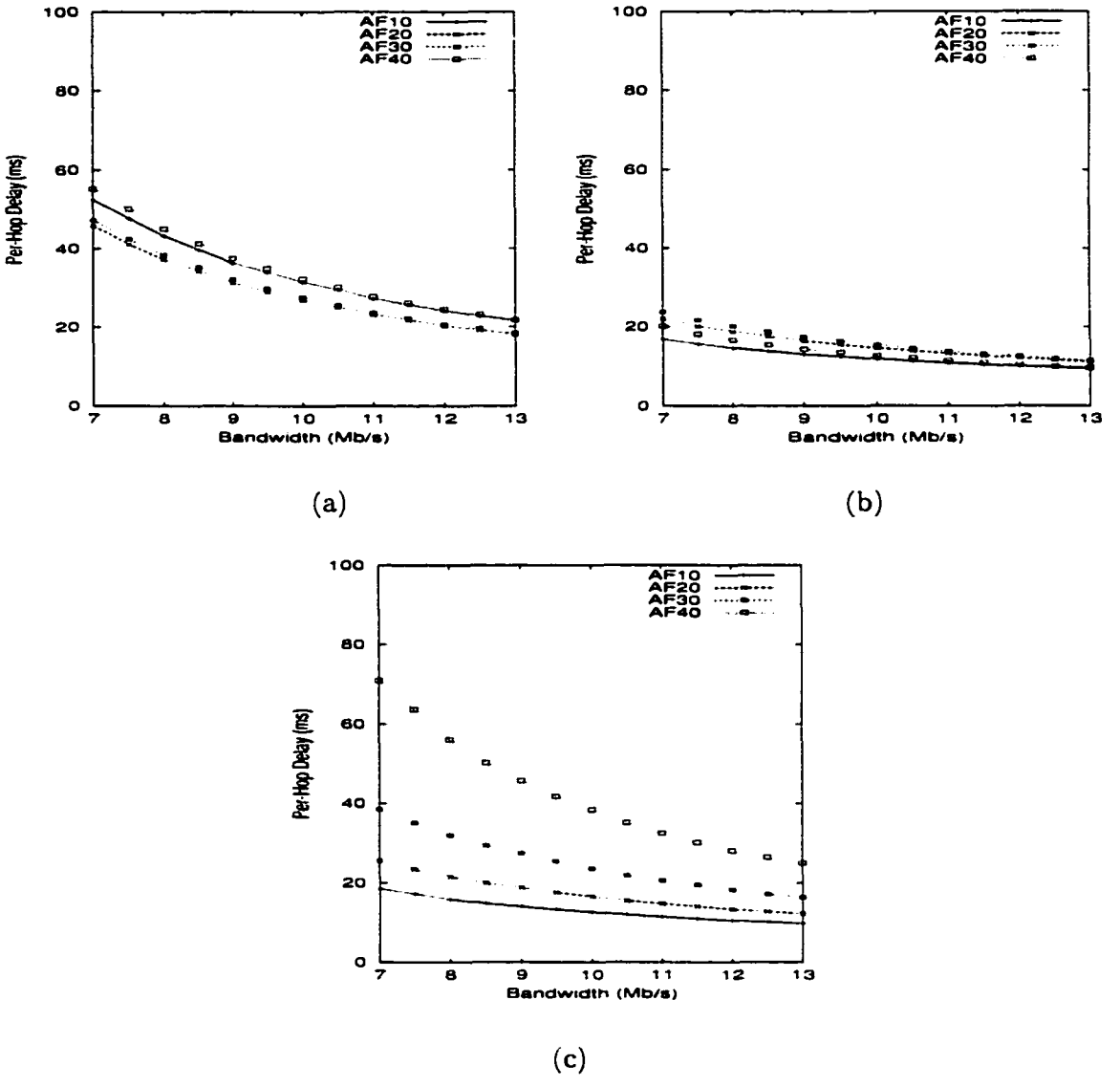


Figure 5.6 Effect of link bandwidth (worst case, unif. distribution) on (a) non-propagation, (b) DSMCast, (c) DSMCast (separate trees)

Table 5.1 Simulation parameters

Parameter	Value(s)
Link Bandwidth, Delay	10.0 Mb/s, 5 ms
DSCPs Uniform Non-Uniform	AFxy (x=1..3,y=1..4) x(25%,25%,25%,25%),y(33%,33%,34%) x(10%,20%,30%,40%),y(15%,30%,55%)
Scheduling	WRR, weights=(24,18,12,6)
Multicast Group(s) Packet Size/Rate	1 group 512 bytes, 125 ms
Background Traffic	50 ms (Inter-Arrival Time) 15 sec (Session hold Time) 1000 bytes, 10 ms 50 % TCP, 50 % UDP

of AF30. Next, R_2 and R_4 join with PHBs of AF10 and AF20, respectively. In the case of the traditional IP multicast graphed in Figure 5.6(a), the PHB is not propagated upstream. Hence, all downstream receivers suffer due to the fact that the AF40 codepoint dominates the multicast tree. In fact, when the PHB of higher priority PHBs is not propagated upstream, the exact opposite behavior of the Good Neighbor Effect is observed.

In contrast, when the higher priority PHBs are propagated upstream, the QoS of the downstream receivers follows that of what would be expected for such services. In the case of DSMCast with a single tree (see Figure 5.6(b)), the join order does not affect the downstream QoS as the PHB is prioritized correctly at the ingress node. The results in Figure 5.6(c) follow a similar trend with the notable lack of benefit of the Good Neighbor Effect to the lower classes.

The performance differences between DSMCast (single tree) and DSMCast (separate trees) clearly show the Good Neighbor Effect. In the separate tree case, there is a clear differentiation between the different classes. In contrast, in the single tree case, the receivers with shared links gravitate towards one another, thus reducing the effect of differentiation. Most notably, the lowest service class (AF40) receives performance similar to the highest service class (AF10) due to its proximity near the higher service class in the network. However, the service differentiation between the classes shows only one side of the problem. Whereas DSMCast with separate trees provides better class differentiation, the net per-hop delay of the single tree case is better for all receivers involved. For the separate tree case, one packet is sent for each receiver as each

Table 5.2 Summary of simulation results (Worst case) - 10.0 Mb/s - Uniform background traffic

Worst Case	Non-propagation IP Multicast (ms)	DSMCast (Single Tree) (ms)	DSMCast (Separate Trees) (ms)
AF10	31.4171	**11.8058**	12.6124
AF20	26.8862	**14.5739**	16.5225
AF30	27.5987	**15.3199**	23.6022
AF40	32.1789	**12.5545**	38.4462

receiver has a different codepoint. In contrast, only a single packet is sent for the single tree case. Thus, the choice between separate trees and a single tree represents an interesting tradeoff, class-wise fairness or network efficiency.

5.3.2 Effect of PHB Propagation (Average case)

Whereas the worst case represents when the non-propagation of upstream PHBs is amplified most, the effect of non-propagation carries a profound effect regardless of join order when considering all cases. Figure 5.7 displays the average case with the same conditions as in Figure 5.6 but with all possible permutations of join order and join location considered. Whereas the non-propagation case improves dramatically from the worst case, the performance on average still does not outweigh the performance of the single DSMCast tree. In fact, the results are even more interesting in the fact that DSMCast incurs a per-packet encapsulation penalty (bandwidth) whereas the traditional IP multicast does not have a per-packet penalty. In addition, it is also interesting to note that on average, while the Good Neighbor Effect does reduce the inter-class differentiation, the performance on average still follows the relative priority of the respective classes.

Tables 5.2 and 5.3 summarize the performance of the respective schemes when the bandwidth is set to 10.0 Mb/s. As the locations of the receivers change, the effects of the background traffic also change, thus representing the increase in per-hop delay from Table 5.2 (worst case) to Table 5.3 (average) for DSMCast.

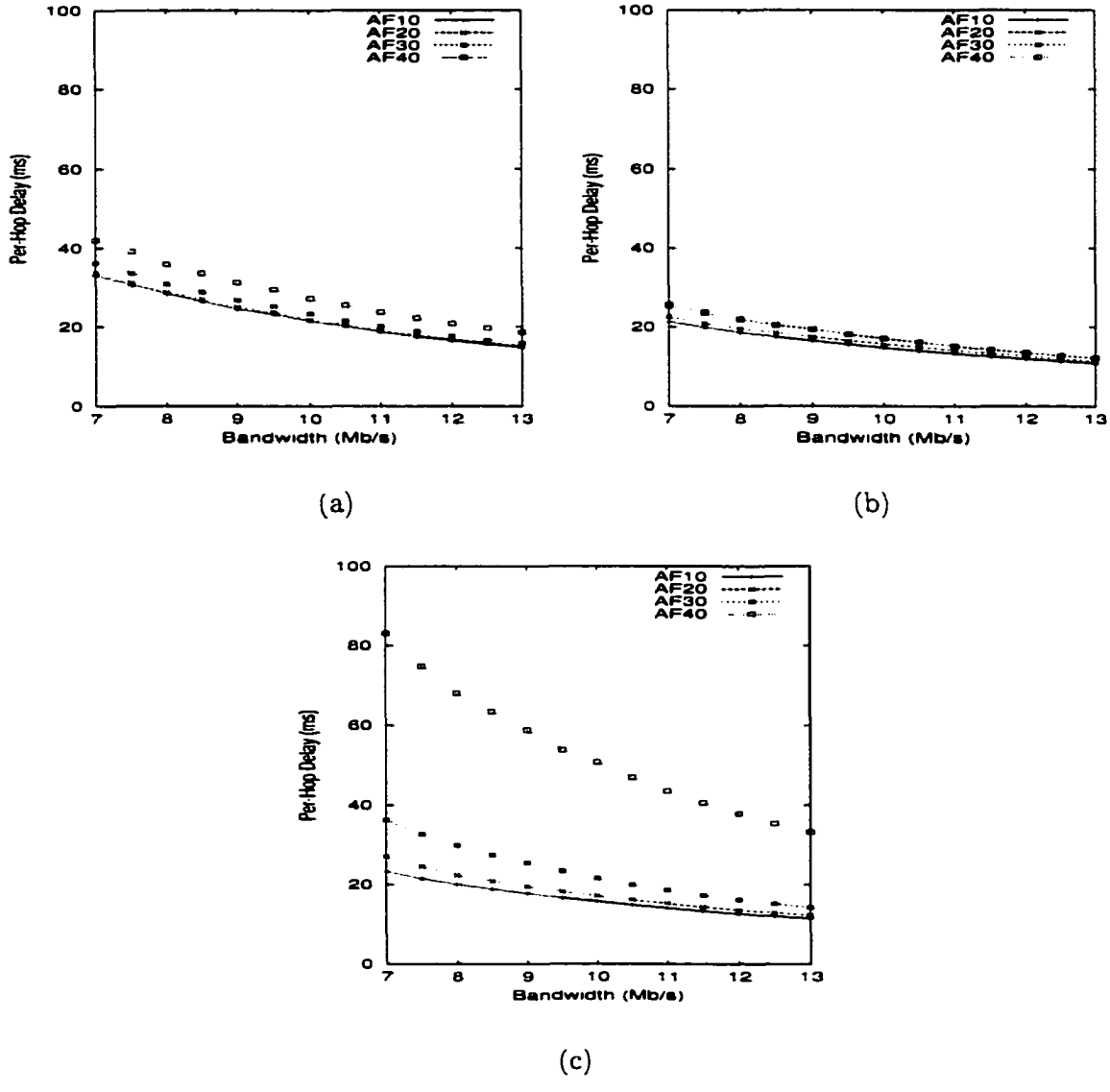


Figure 5.7 Effect of link bandwidth (average, unif. distribution) on (a) non-propagation, (b) DSMCast, (c) DSMCast (separate trees)

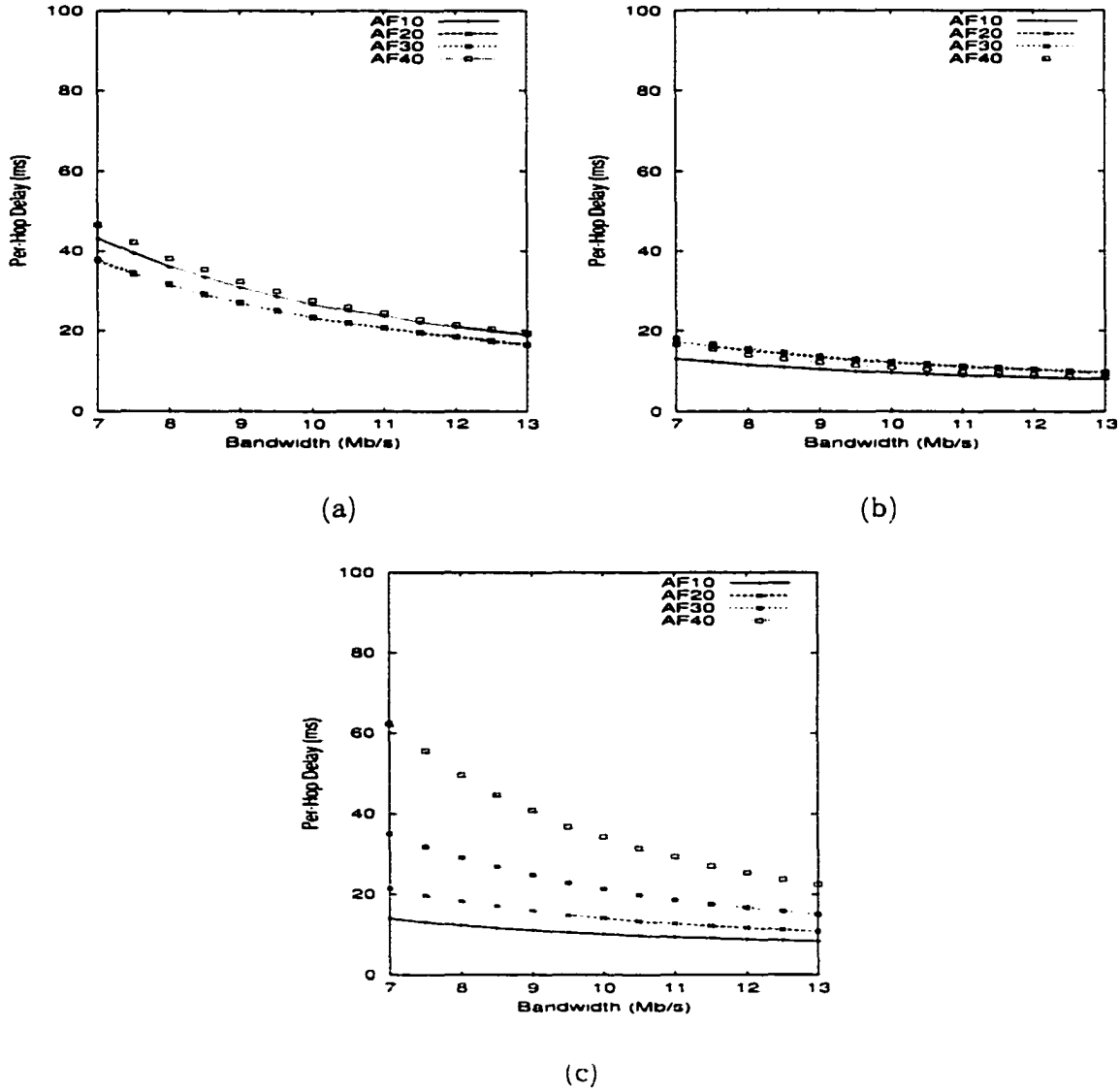


Figure 5.8 Effect of link bandwidth (worst case, non-unif. distribution) on (a) non-propagation, (b) DSMCast, (c) DSMCast (separate trees)

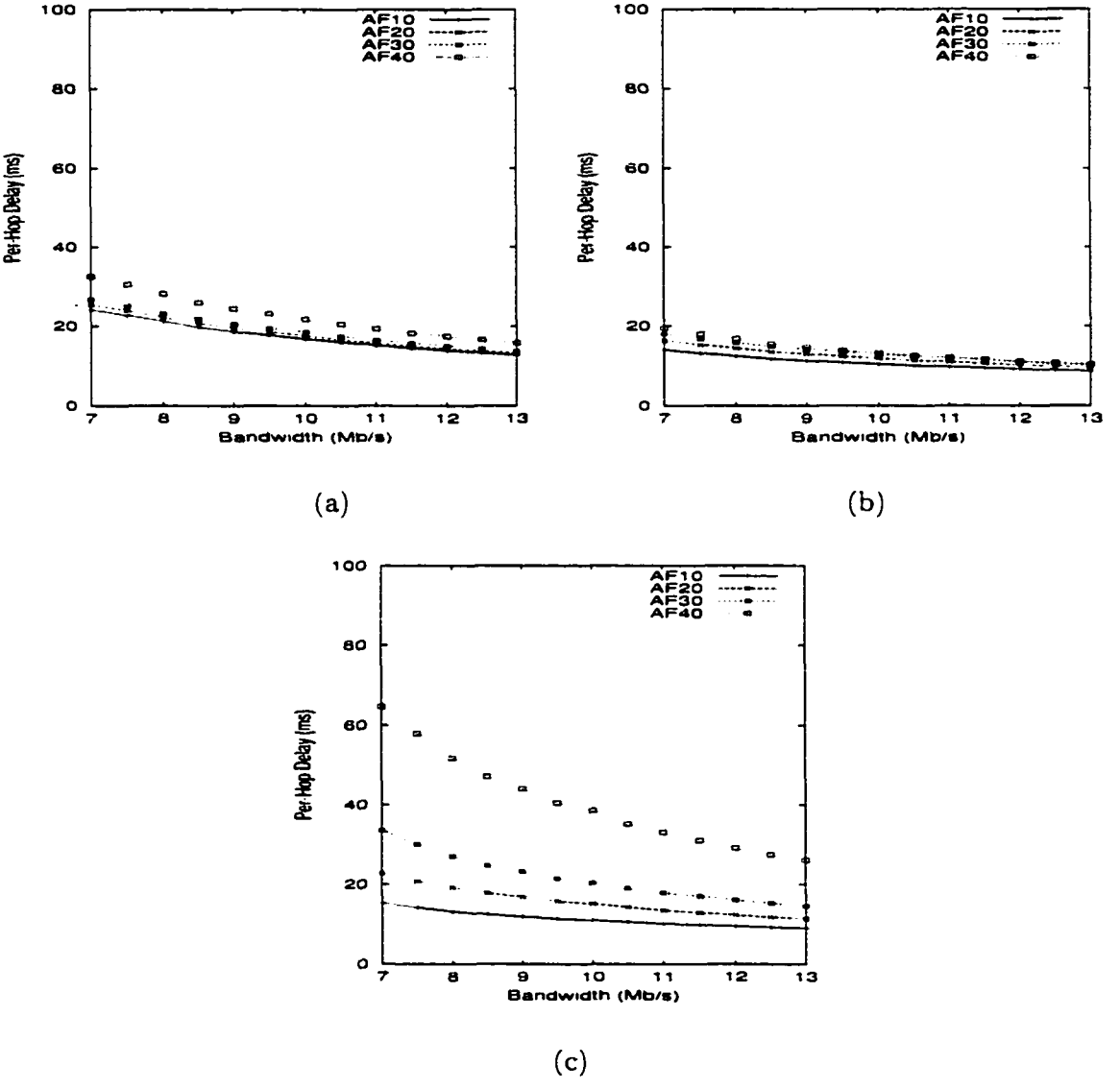


Figure 5.9 Effect of link bandwidth (average. non-unif. distribution) on (a) non-propagation. (b) DSMCast. (c) DSMCast (separate trees)

Table 5.3 Summary of simulation results (Average) - 10.0 Mb/s - Uniform background traffic

Average	Non-propagation IP Multicast (ms)	DSMCast (Single Tree) (ms)	DSMCast (Separate Trees) (ms)
AF10	21.4264	**14.7507**	15.8166
AF20	21.8724	**15.6604**	17.1748
AF30	23.3238	**16.9913**	21.6151
AF40	27.1854	**17.0713**	50.7451

Table 5.4 Summary of simulation results (Worst case) - 10.0 Mb/s - Non-uniform background traffic

Worst Case	Non-propagation IP Multicast (ms)	DSMCast (Single Tree) (ms)	DSMCast (Separate Trees) (ms)
AF10	26.6277	**9.7112**	10.086
AF20	23.2809	**12.1620**	14.0952
AF30	23.5877	**12.5193**	21.3072
AF40	27.6511	**10.9427**	34.3171

5.3.3 Effect of Non-Uniform Traffic

Figures 5.8 and 5.9 show the effect of non-uniform background traffic on the three schemes. In the non-uniform case, the non-propagation of upstream PHBs still has a profound effect for not only the worst case (see Figure 5.8) but also on average as well (see Figure 5.9 and 5.9). However, the upstream propagation effect is reduced from the uniform case as the traffic is concentrated more heavily in the lower priority classes. As a result, less traffic exists with higher priority and hence, the traffic receives less queuing delay. In addition, since each of the four receivers use the highest priority in terms of packet drops ($AFx0$), each of the downstream receivers benefit as more of the background traffic is dropped before the multicast group traffic.

Although somewhat reduced from the uniform case, the Good Neighbor effect is still profoundly evident in the non-uniform case. The effects of the Good Neighbor Effect are evident

Table 5.5 Summary of simulation results (Average) - 10.0 *Mb/s* - Non-uniform background traffic

Average	Non-propagation IP Multicast (ms)	DSMCast (Single Tree) (ms)	DSMCast (Separate Trees) (ms)
AF10	16.7762	**10.4709**	10.9546
AF20	17.5078	**11.8743**	15.1118
AF30	18.5041	**12.9044**	20.3486
AF40	21.7637	**13.2580**	38.543

for the lower priority classes (AF30, AF40) in both the worst case (see Figure 5.8(b)) and on average (see Figure 5.9(b)). Tables 5.4 and 5.5 summarize the performance of the respective schemes when the bandwidth is set to 10.0 Mb/s.

5.4 Conclusions

In this chapter, the effects of heterogeneous QoS in a single multicast tree DiffServ environment were investigated. The use of a single multicast tee presents several interesting dilemmas to the service provider. The first dilemma, PHB prioritization, is directly dependent upon the underlying PHB mechanisms deployed in the DiffServ network and is an open area for implementation research. The second dilemma regarding the value of differentiation versus bandwidth savings is an additional topic for debate. Whereas a single tree offers better bandwidth utilization and per-class performance, the differentiation of the single tree is drastically reduced as opposed to the separate tree. The *Good Neighbor Effect* plays a distinct role in single tree management of heterogeneous QoS and the user QoS effect must be weighed in addition to the underlying resource concerns.

CHAPTER 6. EI-HELLO: EXPEDITING FAULT DETECTION

In this chapter, a novel application of DSMCast, EI-HELLO (Edge-based Intelligent HELLO), is proposed for expediting the fault detection rates of link state protocols. The EI-HELLO model offers significant improvements in terms of router CPU usage and offers performance close to that of the optimal link state approaches. This chapter describes the EI-HELLO model and investigates the EI-HELLO model through extensive simulation studies.

The chapter is organized as follows. Section 6.1 discusses the link state routing problem in detail and presents the network and fault model for EI-HELLO. Next, Section 6.2 details the EI-HELLO model and additional implications of EI-HELLO. Section 6.3 extends the original EI-HELLO model by proposing a novel algorithm for managing the message overhead of EI-HELLO on the network. Then, Section 6.4 examines the performance of EI-HELLO on both small and large network topologies. Finally, Section 6.5 finishes the chapter with several concluding remarks.

6.1 Problem and Motivation

Although the proper provisioning and monitoring of QoS is critical for next generation applications, an equally important factor for QoS is the underlying fault-tolerance of the network. In order to provide the necessary QoS levels to QoS-based applications, the network must offer satisfactory connectivity across the network at all times. The notion of satisfactory connectivity implies that in the event of a fault in the network, the network will recover from the fault before the QoS of the applications using the network is violated. However, the fault detection times of most IP networks and the associated link state protocols are significantly slower than tolerated by most strict QoS applications. In fact, it was observed in [94] that

such slow fault recovery is typically the norm rather than a rarity, hence resulting in typical recovery times on the order of the tens of seconds.

In addition, a second aspect of network connectivity is the impact on resource management. In order to correctly manage resources via policing/shaping, it is critical that the policing/shaping mechanism have a precise picture of the resource impact of the incoming packets on the network. The calculated impact of the resources is dependent upon the predicted route of the packets through the network. The quality of this prediction is directly dependent upon the accuracy of the network topology (network state). Thus, while the QoS management mechanisms (reservation, scheduling, policing, and shaping) are responsible for providing the actual QoS, it is the task of the underlying routing protocols (IS-IS [81], OSPF [80], BGP [95], etc.) to provide an up to date network state for the QoS mechanisms to make accurate decisions.

In DiffServ, the routing state is especially critical as core statelessness prevents per-flow monitoring and flow-based fault handling by the core (see Figure 6.1). In addition, the sender-driven nature of DiffServ requires state information be propagated back to the ingress node or BB (Bandwidth Broker) in order to ensure correct provisioning/differentiation. The problem is further complicated by the distributed nature of the edge nodes in the DiffServ domain. Due to the inherent distributed nature of the network state information, a tradeoff exists between the accuracy of the network state information and the impact of routing protocol on the network (network bandwidth and CPU overhead). On one end of the spectrum, one can keep the network burden low at the cost of a potentially inaccurate network state. At the other end of the spectrum, a more accurate network picture can be achieved at the cost of significant network resources.

6.1.1 Motivation

The tradeoff of the underlying routing protocol introduces the motivation for this chapter. A natural application of the previous work with DSMCast is to propose a model for capturing this tradeoff. Whereas the DiffServ architecture poses unique challenges, it also provides

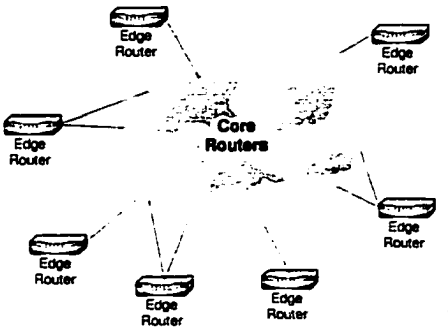


Figure 6.1 Edge-based detection

unique opportunities to take advantage of the underlying DiffServ infrastructure. This chapter proposes a model, EI-HELLO (Edge-based Intelligent HELLO), that uses DSMCast to augment the existing link state protocols through edge-based monitoring of the internal core routers. By employing the concept of a heartbeat data packet and a hybrid HELLO mode, the method is able to augment the performance of link state routing to approach that of a high overhead millisecond-level HELLO model while drastically reducing the amount of core router CPU processing required.

6.1.2 Network Model and Fault Model

For the EI-HELLO model, it is assumed that routers within the domain use link state protocols (such as OSPF [80] or IS-IS [81]) to exchange routing information. The inter-domain routing case is not considered as DiffServ is only interested in packets crossing from one edge (ingress router) of the domain to the other edge (egress router). In general for link state protocols, a HELLO message is sent to all of the router's neighbors in order to test the health of the links. Upon successful receipt of a HELLO message, a router can be certain that packets are flowing correctly from its neighbor to the router[1].

Unless the routers have an error notification mechanism from lower layers (such as in Packet-over-SONET interfaces [96]), the router is forced to rely on HELLO timeouts to determine that a link is down. Timeouts are specified typically by the HELLO interval (H) and a dead time

[1]Note: This does not ensure that packets can flow the other direction (i.e. out of the router to the neighbor).

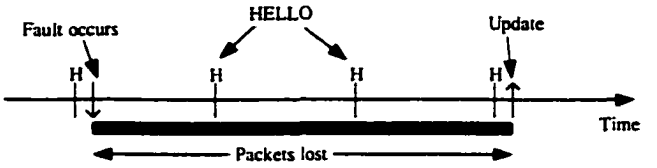


Figure 6.2 Link state fault detection time

(D) where the timeout is simply $H \times D$ seconds (see Figure 6.2). When a topology change is detected, the updated link state is flooded to the entire domain (or appropriate sub-area). If the HELLO interval is large (i.e. 5-10 seconds), the re-route time may be extremely slow. As noted earlier such slow re-route times are typically the norm rather than a rarity, hence resulting in re-routing times on the Internet in the tens of seconds [94].

The simplest solution is to reduce the HELLO interval to a much more acceptable level in order to detect faults faster. Under the current specifications of OSPF and IS-IS, the HELLO interval is limited to an integer value (in seconds). In order to improve the fault detection time further, it was proposed in [97] to add an extension field to IS-IS for millisecond HELLO intervals. For highly delay and loss sensitive traffic such as Voice over IP (VoIP), such a recovery time is essential [94].

However, as discussed earlier, such a reduced HELLO interval does not come without a cost. In addition to the additional bandwidth cost, the processing of additional HELLO messages consumes valuable router CPU resources as well. The CPU cost is due to the fact that the routing protocols are typically not supported in hardware, partly due to complexity and partly due to their predicted relatively low impact[2]. In heavily loaded networks that fail fairly infrequently, such router resources might better be utilized for other operations.

In fact, a DiffServ network is more concerned with simply verifying that an edge-to-edge path is still operating correctly (fault detection) rather than verifying the health of individual links. As a result, a data packet targeted to a specific path (route-pinned or using IP routing) can offer a zero CPU impact method for verifying edge-to-edge connectivity. Thus, assuming

[2]The low impact is based on the default settings for HELLO settings (5-10 seconds).

that data packets consume zero CPU resources and HELLO packets consume significant CPU resources, a hybrid method employing efficient data packets for detecting faults could offer significant router CPU savings.

In fact, a further extension of using data is to verify not only the health of a single route but the health of multiple routes simultaneously. For such an approach, multicast provides the best transport mechanism as it minimizes the bandwidth consumption. By using multicast (DSMCast) for replication and pinning the route of the packet, the edge-to-edge integrity of critical paths in the DiffServ domain can be verified through successful receipt of such packets. In addition, since the packet is a data packet, core routers do not have to process the packet beyond the replication information (implemented in hardware), thus adding zero overhead to the CPU of the router.

6.1.3 Fault Model

In this chapter, it is assumed that the underlying domain is running a link state protocol such as OSPF or IS-IS. The link state protocol detects changes in link topologies through the use of HELLO messages. A link is determined to be *down* if a HELLO message has not been received in $H \times D$ where H (HELLO interval) is the amount of time between successive HELLO messages and D (the dead time) is an integer greater than 0. If a change in the link status is detected (link goes down or link is reactivated), an update message is flooded to all of the nodes in the domain.

The type of faults that may or may not occur in the network are defined in the following manner:

- *Single link failure:* For a duplex link between node A and node B, only (A, B) or (B, A) fails.
- *Full link failure:* For a duplex link between node A and node B, both links (A, B) and (B, A) fail.
- *Node failure:* For a node A in the network, all links outgoing from A fail.

- *Congestion:* Dropping of packets due to congestion is not considered to be a failure in the network and is considered a normal byproduct of network operation.
- *Byzantine failure:* Nodes are assumed to not enter a Byzantine state.
- *Malicious failure:* The routing state protocol is assumed to employ secure routing to prevent malicious (user-initiated) routing failures (replay, man-in-the-middle, etc.).

For multicast packets traversing the DiffServ domain, the route is pinned via information in the DSMCast header. By virtue of pinning the route within the packet header, such an approach is susceptible to faults within the underlying network. If the underlying link state protocol is slow to detect the fault, packets are not routed around the fault until the link state protocol updates the ingress router. However, the route pinning that makes DSMCast susceptible to slow link updates also provides the mechanism for detecting possible faults. Since the route of the packet is known, the successful receipt of packets can be used to produce a mechanism for detecting possible faults.

6.2 EI-HELLO: Edge-based Intelligent HELLO

In this section, an efficient fault detection method is proposed (EI-HELLO) that capitalizes on the unique aspects of the DiffServ architecture (intelligent edge routers and route-pinned multicasting) to monitor the health of internal core nodes in the network. Rather than greedily operating with a fast HELLO interval (sub-second) all the time, the EI-HELLO method engages the sub-second HELLO interval (*hybrid HELLO*) only when a possible fault is detected. Potential faults are detected by the edge routers through the use of *heartbeat* packets (specific route-pinned data packets) sent across specified links. The edge routers then use the distributed feedback of other edge routers and internal timers to determine the appropriate circumstances for the fast HELLO intervals. The proposed EI-HELLO method can be divided into two main components, the *heartbeat* component (fault detection) and the *hybrid HELLO* component (fault location + fault recovery).

- *Heartbeat*: A heartbeat-style approach is used to allow the edge routers to periodically verify the health of specified links in the network without the CPU overhead of HELLO messages. The heartbeat component is not responsible for pinpointing the exact location of the fault, rather it is only responsible for detecting that a fault may be present in the network. In order to avoid additional CPU overhead or the special treatment of control packets, the heartbeat is sent as a data message with a minimal packet loss probability. The heartbeats are processed in a distributed manner by edge routers to determine the health of links in the network.

- *Hybrid HELLO*: When an edge router detects a possible fault, the edge router will selectively switch suspect core routers into a hybrid HELLO mode. In the hybrid HELLO mode, the core router will send HELLO messages to the specified neighbors (or all neighbors) at a significantly increased rate (sub-second interval) for a specified number of HELLO packets. When the specified number of hybrid packets is complete, the router will switch back to normal HELLO message operation (multiple second interval).

6.2.1 Edge-based Heartbeats

As mentioned earlier, core routers are not allowed to maintain any state information besides per-class state information. Thus, the monitoring of the successful delivery of multicast or unicast packets is not possible via state information in the core. From the perspective of the edge router, the sender-driven QoS makes it desirable to be able to verify the health of the route to the egress nodes. Whereas per-flow monitoring would offer the best monitoring capabilities, both class-wise monitoring and simple edge-to-edge connectivity monitoring could offer potential benefits.

For example, if an edge router could monitor the congestion in the core across a specific path for a range of classes, the edge router would be able to offer an adaptive packet marking scheme. In such a scheme, the edge routers could try to intelligently minimize the network cost while appropriately meeting the edge-to-edge QoS requirements. The impact of EI-HELLO on QoS management is discussed in more detail in Section 6.3. In the case of route-pinned

packets (such as with MPLS [98] and DSMCast), a simple verification of connectivity is key as such packets are not locally rerouted in the core.

In order to accomplish the goal of external health monitoring, the edge router will issue a special *heartbeat* data packet to the other edge routers in the DiffServ (DS) domain. The route-pinned approach of DSMCast offers the most efficient method for accomplishing this goal. If the multicast tree for the heartbeat packet contains all of the paths employed by a given edge router for data packets to all other edge routers, a successful acknowledgement of the heartbeat packet from all of the edge nodes means that the links used by the edge node to transmit data to the other edge nodes are functioning. However, such a condition does not imply that all of the links in the entire DS domain are operating, rather only that packets are simply being routed successfully from a given edge node to the other edge nodes along the verified paths.

Formally, consider an edge node E_{Ing}, an ingress node in the DS domain that wishes to verify that packets are flowing along valid paths (no links or nodes are down) to the egress (downstream) edge nodes. Assume that E_{Ing} knows the complete path from edge-to-edge across the domain[3]. Let $E_1, \ldots, E_N$ represent the egress nodes receiving packets from E_{Ing} and let $L_1, \ldots, L_X$ represent all of the links traversed by the packets. Thus, the problem can be summarized as:

Construct a set of multicast trees ($T_1, \ldots, T_X$) such that for each L_i, L_i is covered by at least one branch of one of the multicast trees.

6.2.2 Simple Case - Single Tree

To start, consider the case where all of the packets from E_{Ing} to other edge nodes can be condensed into a single multicast tree. In this case, only one potential path exists for each ingress/egress pair. In such a case, it may be possible that the paths may not be the same in the forward and reverse directions (i.e. $P(E_x, E_y)$ may be different from $P(E_y, E_x)$). In short, the multicast tree would contain all paths for all data packets used by E_{Ing} for all

[3]For flows that have reserved resources, such paths would be available from the BB (Bandwidth Broker). For other flows, the link state routing protocol would provide such paths.

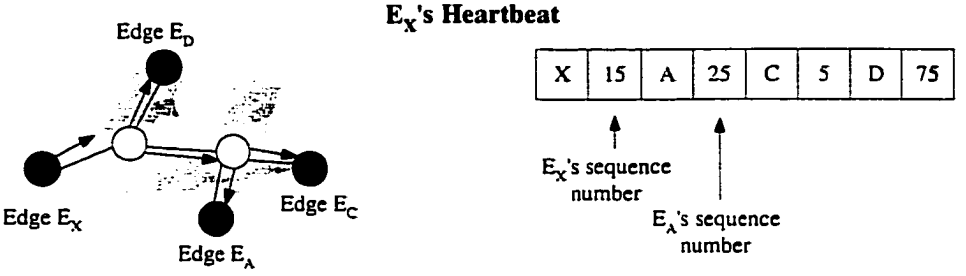


Figure 6.3 EI-HELLO heartbeat exchange

outbound packets. Heartbeat packets would be transmitted at an interval of *HBI* (heartbeat interval) seconds across the multicast tree and would be uniquely identified through the use of a sequence number.

Although the successful receipt of a heartbeat packet at E_i from E_{Ing} denotes that the path $P(E_{Ing}, E_i)$ is working, the packet does not necessarily imply that $P(E_i, E_{Ing}))$ is valid and thus is only marginally useful. As a result, the downstream node (E_i) would need to acknowledge the successful receipt of the packet to the ingress node (E_{Ing}) since the ingress node is the node actually concerned with the validity of $P(E_{Ing}, E_i)$.

In order to avoid the problem of message explosion, the heartbeat and acknowledgement information are consolidated into a single packet. Rather than acknowledging individual packets, an edge node simply includes the sequence numbers from all known edge nodes in the heartbeat packet (see Figure 6.3). Since the heartbeat is transmitted to all other edge nodes, the sender effectively acknowledges all other edge nodes with a single packet.

As a result of receiving an acknowledgement from E_i, E_{Ing} can be reasonably certain that data packets are correctly being sent to E_i if the heartbeat packet from E_i contains the correct sequence number for E_{Ing}. In such an instance, the paths used by the multicast distribution tree between E_{Ing} and E_i as well as in the reverse direction can be assumed to be correct[4] if the sequence numbers are synchronized. Note that each edge node has its own independent sequence number for its transmitted heartbeats.

[4]This assumes that the neither E_{Ing} nor E_i is malicious nor that heartbeat packets are being spoofed by an attacker.

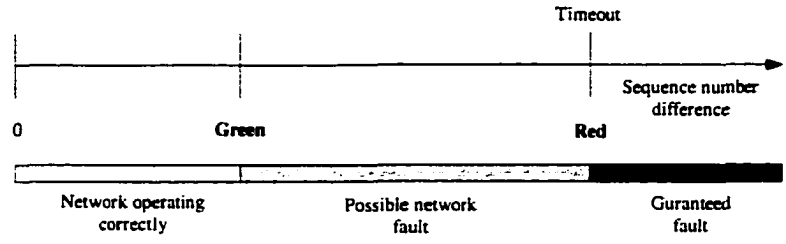


Figure 6.4 EI-HELLO Red and Green parameters

Since the round-trip delay across the domain may exceed the interval between when heartbeat packets are sent, heartbeats are allowed to be out of sequence by a fixed amount. In order to characterize such tolerances, the EI-HELLO model has two parameters for controlling the inconsistent state of the sequence numbers, namely the *Red* and *Green* parameters. The *Red* and *Green* parameters are given as multiples of the heartbeat interval (*HBI*). As the names imply, the *Green* parameter implies that an edge router is receiving packets while *Red* implies that there is a potential fault in the network. Figure 6.4 shows how the *Red* and *Green* parameters correspond to the operation of EI-HELLO.

6.2.3 Red Parameter

The *Red* parameter controls the timeout and is used to detect full link or node failures between two edge routers. In such a case, heartbeats are not reaching their respective targets in either direction. From the perspective of E_{Ing}, it will notice that it has not received a heartbeat message from E_i in $Red \times HBI$ seconds where *Red* is an number greater than zero and *HBI* is the interval between successive heartbeat messages. In order to reduce the number of timers, the status could be checked only on the periodic interval. Although E_{Ing} does not know which link between E_{Ing} and E_i is down. E_{Ing} can assume with reasonable certainty that a link is probably down (i.e. not transient congestion). In order to avoid false negatives, the *Red* value should be set sufficiently high as well as using an appropriate PHB to prevent excessive dropping of heartbeat packets. In the case that such a node is discovered, the hybrid HELLO routine should be invoked to probe the path from E_{Ing} to E_i. The hybrid HELLO

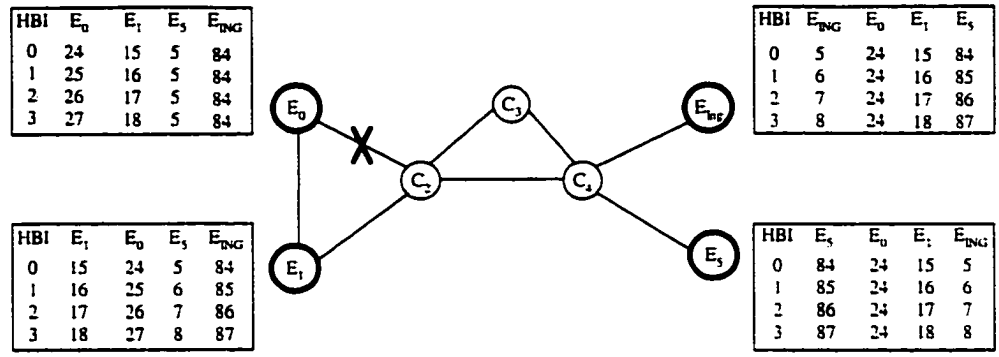


Figure 6.5 EI-HELLO heartbeat warning analysis

routine is discussed in greater detail in Section 6.2.5.

6.2.4 Green Parameter

While the *Red* parameter will detect faults via a timeout, the *Green* parameter plays a much more critical role in the operation of the EI-HELLO model. The *Green* parameter is responsible for detecting single link failures and is used by other edge nodes to assist in detecting link failures. The intuition behind the *Green* parameter relies on the distributed intelligence of the edge routers to detect potential failures before the *Red* timeout.

Consider a network such as the one pictured in Figure 6.5 where E_{Ing} is sending heartbeats to the three other edge nodes, E_0, E_1, and E_5. Suppose the link between nodes C_2 and E_0 goes down (both directions). In this case, the heartbeat message between E_{Ing} and E_0 will not be received successfully. However, E_1 will still receive heartbeats from E_0 due to the direct link between them. As a result, E_1 will notice that E_0's picture of E_{Ing}'s sequence number is out of sync with the sequence number that E_1 has for E_{Ing}. Without any intervention, the *Red* timeout would eventually occur and proper probing and recovery would soon follow.

However, since E_1 recognizes that E_0 is behind by more than *Green* sequence numbers, E_1 can attach a warning to its next heartbeat letting all of the edge nodes know that there is a potential problem between E_0 and E_{Ing}. Since the heartbeat message is already going between E_1 and the other nodes, it is more efficient to attach a warning to the heartbeat message rather

than separately unicasting to E_0 and E_{Ing}.

Even if E_1 does not detect that E_0 is behind in the heartbeat sequence, E_0 could also use the heartbeat from E_1 to detect that it is out of sequence with E_{Ing}. E_{Ing} could also use the same procedure with the heartbeat from E_1 as well for E_0's sequence number. Thus, although an edge node may be blocked from receiving a heartbeat, it may attempt to recover before the *Red* timeout as long as it is still receiving heartbeats from other nodes. The key concept for the *Green* parameter and warning is that although one link or node may fail, the failure can be detected much sooner if connectivity to the edge node still exists otherwise.

6.2.5 Hybrid HELLO

Once a possible fault has been detected, an edge node needs to trigger the network to try to locate/recover from the possible fault. Although the edge node could attempt to probe the network itself via various probing packets to/from core nodes, such a process would be wasteful and time-consuming. Since the link state protocol is already responsible for appropriately detecting faults, the edge node can engage the routers along the suspected path to attempt to detect potential faults.

Although the fault will eventually be recovered from by the link state protocol, the re-routing time may be excessively long. Thus, since the edge node knows the path(s) containing the potential fault, the edge node can trigger the routers on the path(s) to enter a hybrid HELLO state whereby the routers reduce their HELLO intervals temporarily in the interest of detecting possible faults.

Figure 6.6 shows an example of a hybrid HELLO activation message. In the figure, E_{Ing} detects a possible fault on the paths to E_0 through a warning message from E_1. As a result, E_{Ing} sends an *Activate-Hybrid* message on a multicast tree containing only E_0 using the same paths as the heartbeat packet. When a node receives the *Activate-Hybrid* message, the node will convert the upstream link as well as the downstream links to the hybrid HELLO mode (according to the replication information in the DSMCast header). The hybrid HELLO mode is active for $HybHC \times HybH$ seconds where $HybHC$ is the number of hybrid HELLO messages

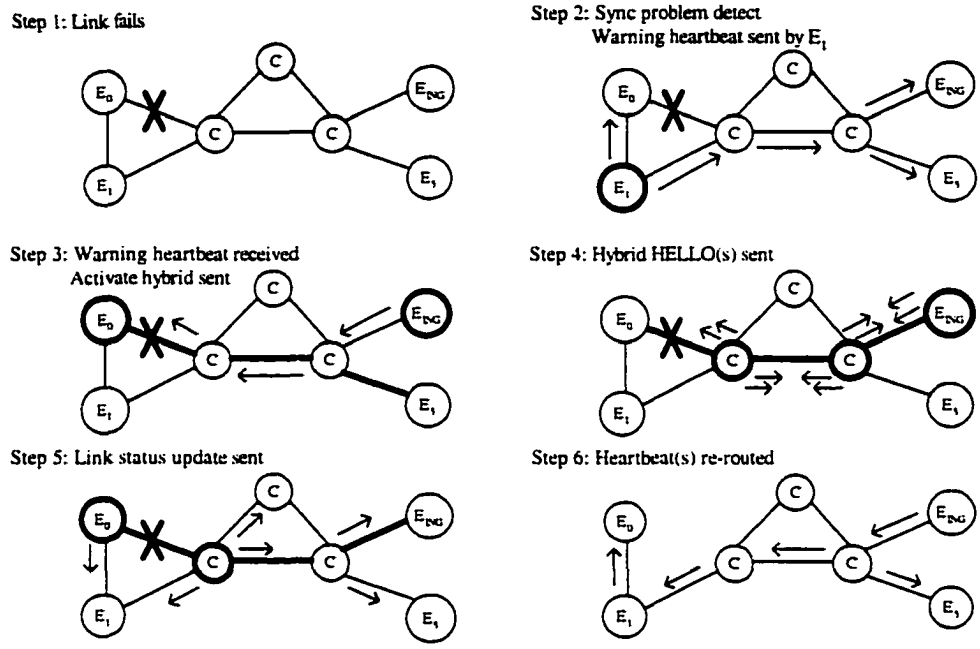


Figure 6.6 Hybrid HELLO Activation

to send and $HybH$ is the temporary HELLO interval to use for the hybrid HELLO packets. During the hybrid HELLO mode. links that are in the hybrid HELLO mode are considered dead after $HybH \times HybD$ seconds where $HybD$ (hybrid dead time) is an integer greater than 0. If the links are functioning correctly, both the upstream and downstream nodes will be converted to the hybrid HELLO mode for the specified timeframe for only those specific links. In the case where such a message does not propagate fully (link/node failure), an error will be detected as the network routing state will be updated.

In addition to E_{Ing} detecting the fault, E_0 would also have received the warning from the heartbeat of E_1. As a result, E_0 would also have sent an *Activate-Hybrid* message towards E_{Ing}. Note that the *Activate-Hybrid* message does not need to reach either E_{Ing} or E_0. Rather, the *Activate-Hybrid* message needs only to reach the node just before the fault. In fact, if the *Activate-Hybrid* reaches its intended destination, it implies that a false positive was detected since the edge-to-edge path is still functioning.

The use of the hybrid HELLO mode has several advantages. First, the link state routing

protocol does not have to use the reduced HELLO interval unless in hybrid mode. For the majority of the network operation, the HELLO interval will be a typically larger value, thus representing a savings in terms of CPU processing time at core routers. Although the heartbeat messages do cost additional bandwidth, the heartbeat packets are treated as data packets in the core and thus cause zero CPU processing overhead. Thus, through the use of the distributed intelligence of the edge routers coupled with the use of heartbeat packets as input, the fast link state interval can be intelligently triggered rather than constantly being used.

6.3 Improving EI-HELLO Performance

In the above descriptions, each edge router employs a greedy approach whereby each edge router attempts to verify all of its respective data paths. However, in sparse networks, such a greedy approach may be unnecessary as nodes could divide the verification tasks among themselves. This is due to the fact that the network sparsity causes many of the verifications to be redundant. Thus, a further consolidation is needed beyond the multicast tree to increase the efficiency of the heartbeat packets. The problem could be summarized thusly:

For a given DS domain (D) containing a set of edge nodes E and a set of core nodes C, construct a set of multicast M trees rooted at nodes in E where each link (X, Y)in D is traversed by a minimal number of heartbeat packets over both (X, Y) and (Y, X).

Although it may be possible to construct an optimal set of trees that minimizes the impact on the network (such as using a Maekwa set [99]), the process will be extremely difficult if not impossible. Thus, the next best choice is to propose a heuristic that attempts to reduce the message impact on the underlying network.

6.3.1 Reducing EI-HELLO Overhead

During correct operation, each edge node needs only receive a heartbeat from the other edge nodes once per *Green* cycle. Rather than greedily verifying the health of the links to all of the other edge nodes, an edge node can simply send a heartbeat to a subset of the edge nodes on each heartbeat. However, one must be careful to still meet the minimum heartbeat interval

(denoted by the *Green* parameter), lest EI-HELLO be plagued by excessive false positives. The core problem can be subdivided into two main questions, how to divide up the edge nodes into subsets and which subsets to verify on each heartbeat.

6.3.2 Choosing Subsets

The primary motivation when grouping edge nodes into clusters (subsets) is to try to accomplish the maximum amount of verification (maximize the number of verified paths to edge nodes) while minimizing the total cost of communication. Thus, since all edge nodes must eventually be verified, the intuition behind the clustering mechanism is to base the clusters on nodes that are close together. By grouping edge nodes that are close together, the chances that the edge nodes will share a common path to the ingress node is increased. A greater number of shared links means that the heartbeat is verifying not just one but multiple paths to egress nodes. While the multicast tree solved this problem originally, the problem of constructing partitioning the multicast tree is a distinctly different problem.

To construct the clusters, a method similar to the ECT algorithm of EBM is proposed. Unlike ECT, all edge nodes are candidates for the cluster as there is no PHB precedence. First, a candidate cluster is constructed around each edge node that contains the edge node and all edge nodes within CH hops. The metric for evaluating the different clusters is derived by dividing the cost of the multicast tree to the cluster by the number of edge nodes in the cluster.

$$M(C_x) = \frac{Cost(T_x)}{|C_x|}$$

where C_x is the cluster centered at node E_x. T_x is the multicast tree to reach all of the edge nodes in C_x, and $|C_x|$ is the number of edge nodes in the cluster. The clusters are selecting by iteratively choosing the best cluster (lowest value) until all edge nodes are covered by a cluster. An edge node can only be part of one cluster and is removed from all other candidate clusters once selected as part of a cluster.

The cluster distribution may be specified in one of two fashions. The network administrator

may specify either a total of X clusters or a hop distance of CH hops. Although both are targeted towards the same end goal, the X cluster case can simply be derived by iteratively stepping through CH until a total number of clusters is achieved that is less than or greater than X.

6.3.3 Scheduling Subsets

The second question to answer is how and when to schedule the cluster (subset) to receive a heartbeat. Although subsets could be scheduled dynamically and the tree constructed on each heartbeat, such a scheme consumes excessive computational resources. In contrast, a static scheduling scheme offers a better approach as the priority of heartbeat transmissions is unlikely to change and the trees themselves (i.e. the egress points) are unlikely to change as well.

In the static scheduling scheme, a tree is constructed based on the clusters for each heartbeat transmission. The schedule is developed for 1 to *Green* and repeated until the tree changes. On each heartbeat, at least T percent of the edge nodes must be addressed by the heartbeat. Clusters are selected for scheduling until at least T percent of the edge nodes are covered by the heartbeat. In the event that a topology change is detected, the schedule and trees are recomputed.

The T parameter captures the tradeoff of false positives versus network overhead. With a T near 0%, the minimal number of heartbeats are sent out resulting in a single cluster being contacted each heartbeat. With a T of 100%, the performance will be the same as the greedy approach of the previous EI-HELLO algorithm. The ideal value for T should be $\frac{1}{Green}$ which causes all of the edge nodes to see one heartbeat per *Green* interval.

6.3.4 Benefits and Tradeoffs

The extension to EI-HELLO offers several benefits and tradeoffs which are discussed below:

- *Reduced overhead:* By employing partial verification on each heartbeat, each edge node can consume significantly less overhead while achieving nearly the same performance.

Since *Green* × *HBI* is the soonest that EI-HELLO can detect a fault, the successful receipt of one heartbeat per *Green* interval will keep EI-HELLO operating normally without the overhead of the greedy approach.

- *False positives:* If the extension is not tuned properly, there is the potential for additional false positives. As a possible fault in the network causes an *Activate-Hybrid* message to be sent and the hybrid HELLO mode to be used, a large number of false positives can cause the overhead of EI-HELLO to increase dramatically. In fact, without proper configuration, EI-HELLO could degrade to a performance worse than the fast HELLO module due to both the hybrid HELLOs and the heartbeat overhead.

6.3.5 Additional Extensions

In addition to the performance improvements, EI-HELLO can also offer benefits for QoS management. As an additional feature, heartbeat packets could be used to gather information regarding the status of the core nodes (queue length for each class, drop rate for each class, etc.). Rather than being computed in real-time, the values inserted into the heartbeat could represent weighted averages that are computed at a regular interval. This approach could offer significant benefit for QoS marking and resource provisioning in a DiffServ network as an edge node would have a better picture of the performance of the core of the network. However, this problem is beyond the scope of this dissertation and is a compelling topic for future research.

6.4 Simulation Studies

In order to evaluate the performance of the EI-HELLO model, extensions to the *ns-2* simulator [90] were developed. A generic link state protocol was created that mimics the common features of IS-IS [81] and OSPF [80] such as the HELLO message and the flooding of link state updates. For the simulations, the performance of four separate models was evaluated, a slow HELLO model, the EI-HELLO model, a fast HELLO model, and a Packet-over-SONET model (for baseline purposes). The characteristics of each model are shown in Table 6.1.

Table 6.1 Simulation models

Parameters	IP-SONET	Slow HELLO	EI-HELLO	Fast HELLO
HELLO Settings	None	5 s	5 s	100 ms
Fault Detection	Lower layer Instant	Dead Time 3 x 5 s	Dead Time+HBeat 3 x 5 s/Hybrid	Dead Time 3 x 100 ms
Heartbeat Rate	None	None	250 ms	None
Green/Red	None	None	2/4	None
Hybrid HELLO	None	None	100 ms/3	None

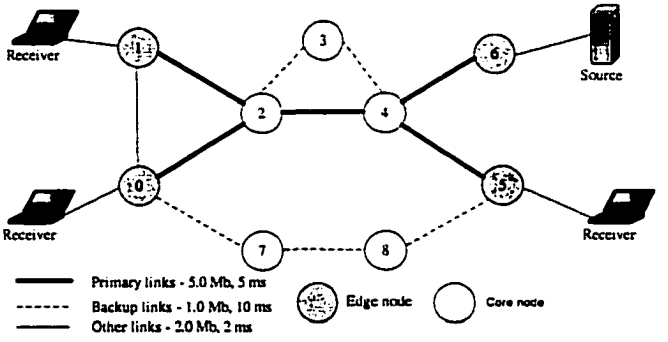


Figure 6.7 Small network topology

The performance of the models were evaluated on two network topologies, a small test network and NSFNet [92]. The small network was selected to isolate the fault detection and recovery times of a single flow. The NSFNet topology was selected as a realistic topology that would be indicative of a practical DS domain.

6.4.1 Small Network Performance

Figure 6.7 details the topology of the DS domain considered in the small network study. For the study, a fault was injected at 10 seconds and Node 0 was monitored for the time until the next data packet (maximum packet gap in seconds) was received. The *maximum packet gap* metric provides an excellent metric for measuring the responsiveness of the failure detection and recovery mechanisms. In addition, a count of *average link state messages* (ALSC) was recorded which measured the average number of link state related messages per node per second. In the

count, a link state related message includes heartbeats, updates, HELLO messages, and hybrid mode activations across each link in the network. For the EI-HELLO model, the overhead of the link state protocol and the heartbeats are shown separately.

6.4.1.1 Link Failure

For Figures 6.8 and 6.9, the link from Node 0 to Node 2 fails at 10 seconds until the end of the simulation (50 seconds). Figure 6.8(a) shows the recovery time as the HELLO interval is varied. The slow HELLO model provides a curve that is fairly indicative of most integer-limited HELLO models. If the default HELLO interval value for the majority of link state protocols is used (10 seconds), the recovery time is on the order of 30 seconds, an unacceptable re-routing time for most flows. However, although a fast HELLO interval offers significantly better re-routing performance, such performance comes at an additional overhead as shown in Figure 6.9(a). In contrast, the EI-HELLO offers performance approaching that of the fast HELLO interval but at a significantly reduced penalty for a lower HELLO interval since the lower HELLO interval is only engaged when a possible fault is detected.

The other figures show the performance of EI-HELLO as the heartbeat interval and *green* settings are varied versus a fixed fast HELLO (100 ms) and a slow HELLO (5 s) interval. Although the heartbeat interval offers some gain for EI-HELLO, the fault detection is limited by the hybrid HELLO rate and *green* setting. Thus, setting the heartbeat interval to a lower level offers only marginal performance gains at a significant cost. Figure 6.9(b) plots the significant cost of EI-HELLO as the heartbeat interval is varied.

The *green* setting in Figure 6.8(c) impacts the performance of the EI-HELLO significantly as the *green* setting determines when other edge nodes can offer warnings regarding synchronization problems of sequence numbers. However, whereas the *green* setting can have a profound impact on the re-routing time, it has very little impact on the message overhead (see Figure 6.9(c)). This arises from the fact the *green* setting is constrained by the end-to-end delay of the network and is used as a threshold rather than for directly generating packets at a fixed interval.

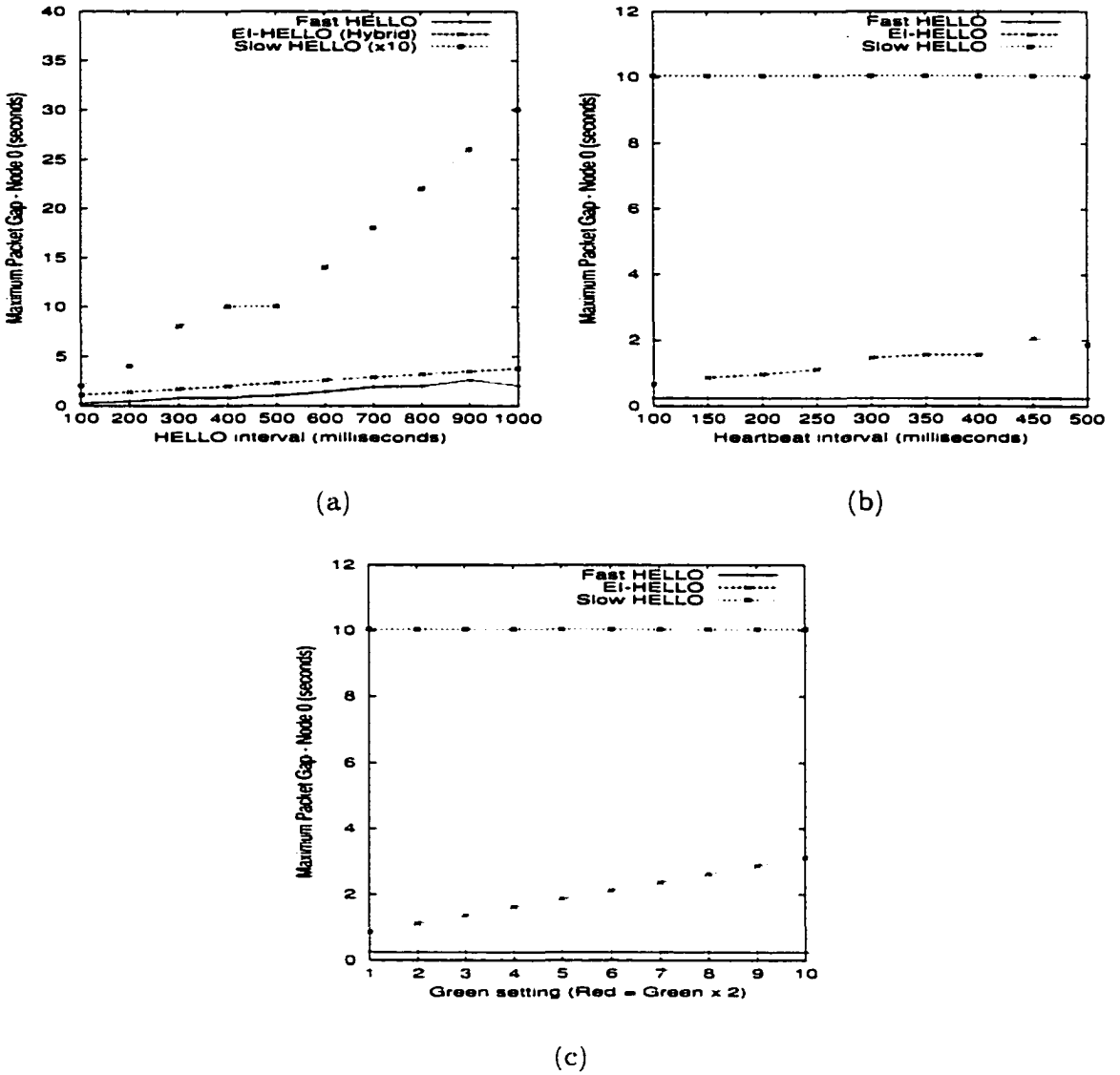


Figure 6.8 Effect of EI-HELLO settings on recovery time - link failure - small network - (a) HELLO interval, (b) Heartbeat interval, (c) Green parameter setting

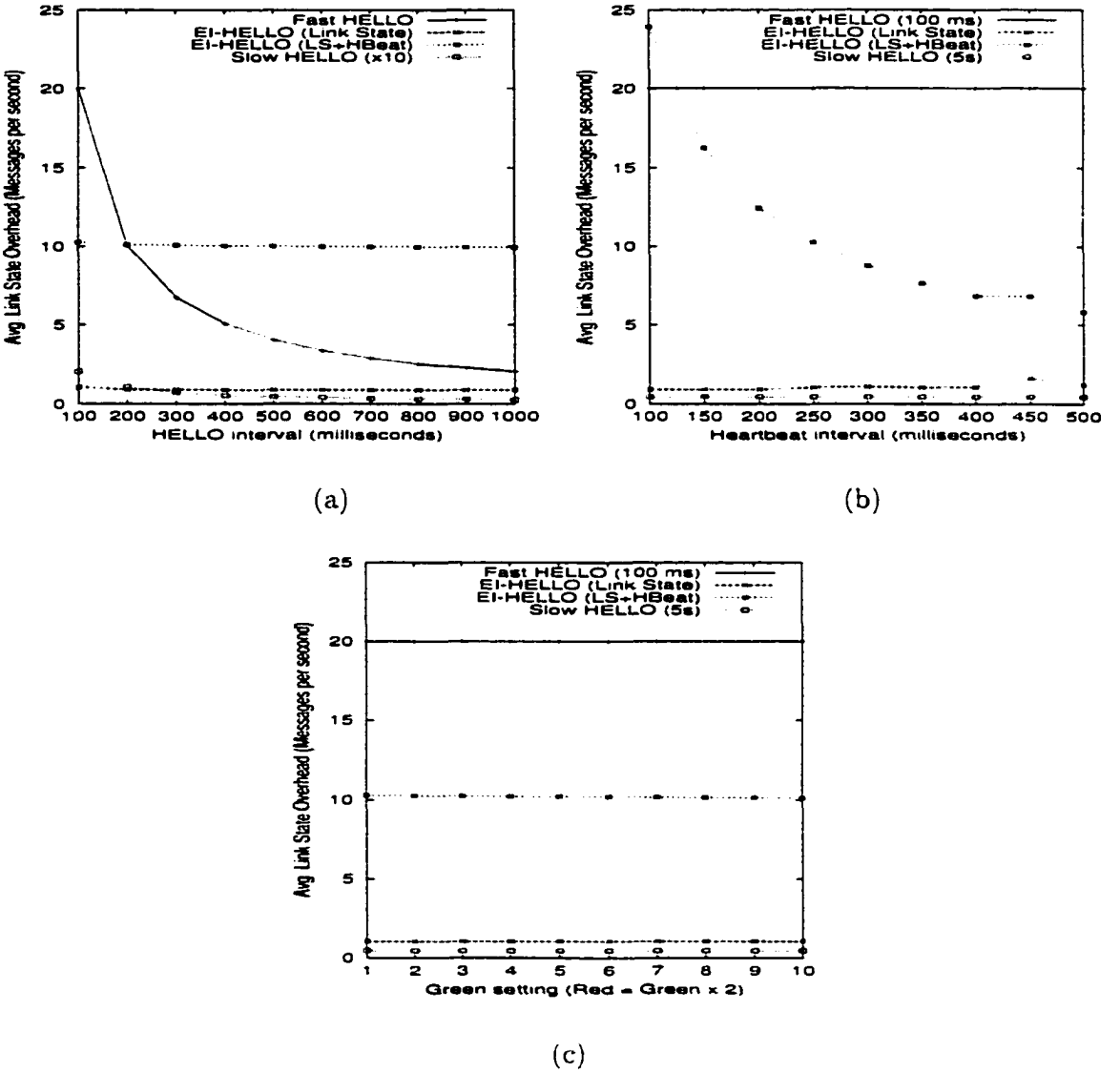


Figure 6.9 Effect of EI-HELLO settings on message overhead - link failure - small network - (a) HELLO interval. (b) Heartbeat interval. (c) Green parameter setting

6.4.1.2 Node Failure

In Figure 6.10, node 2 fails at 10 seconds until the end of the simulation. In the event of a node failure, the performance of the EI-HELLO model follows a similar trend as the one seen in the single link failure case. However, whereas in the link failure case node 1 was able to offer warnings to node 6, the complete failure of node 2 prevents either node 1 or node 0 from receiving heartbeat packets. Thus, the *red* setting (twice the *green* setting) provides the triggering mechanism for engaging the hybrid HELLO interval. As a result, the re-routing time for EI-HELLO is increased while both the fast and slow HELLO modes remain roughly the same.

6.4.2 Large Network Performance

In order to evaluate the performance of the EI-HELLO model on a larger scale, the performance was analyzed on the NSFNet topology [92]. The topology provides a realistic domain to measure the effect of failures in a DS domain. The topology consisted of 33 links, 17 edge nodes, and 13 core nodes with only a single link from an edge router to the core routers. For the simulations, the following parameters were used:

- 38 multicast groups were employed with an average of 4 receivers per group. Member join and leave events occurred on an average of 250 ms apart.
- Faults were exponentially distributed with a MTTF (Mean Time to Failure) of 200 seconds and a MTTR (Mean Time to Repair) of 50 seconds. Both node as well as link faults were injected with equal MTF and MTF characteristics.

Although the fault rate is significantly higher than would occur in practice, the fault rate has been increased to demonstrate the recovery performance of the models. Hence, the slow HELLO interval was set to 5 seconds rather than the default of 10 seconds as with most routers. Since packets are actually being simulated in ns, using realistic fault rates would require vast amounts of computational resources. Multicast packets were selected as the effect of faults on route-pinned multicast (DSMCast) is significantly higher than unicasting.

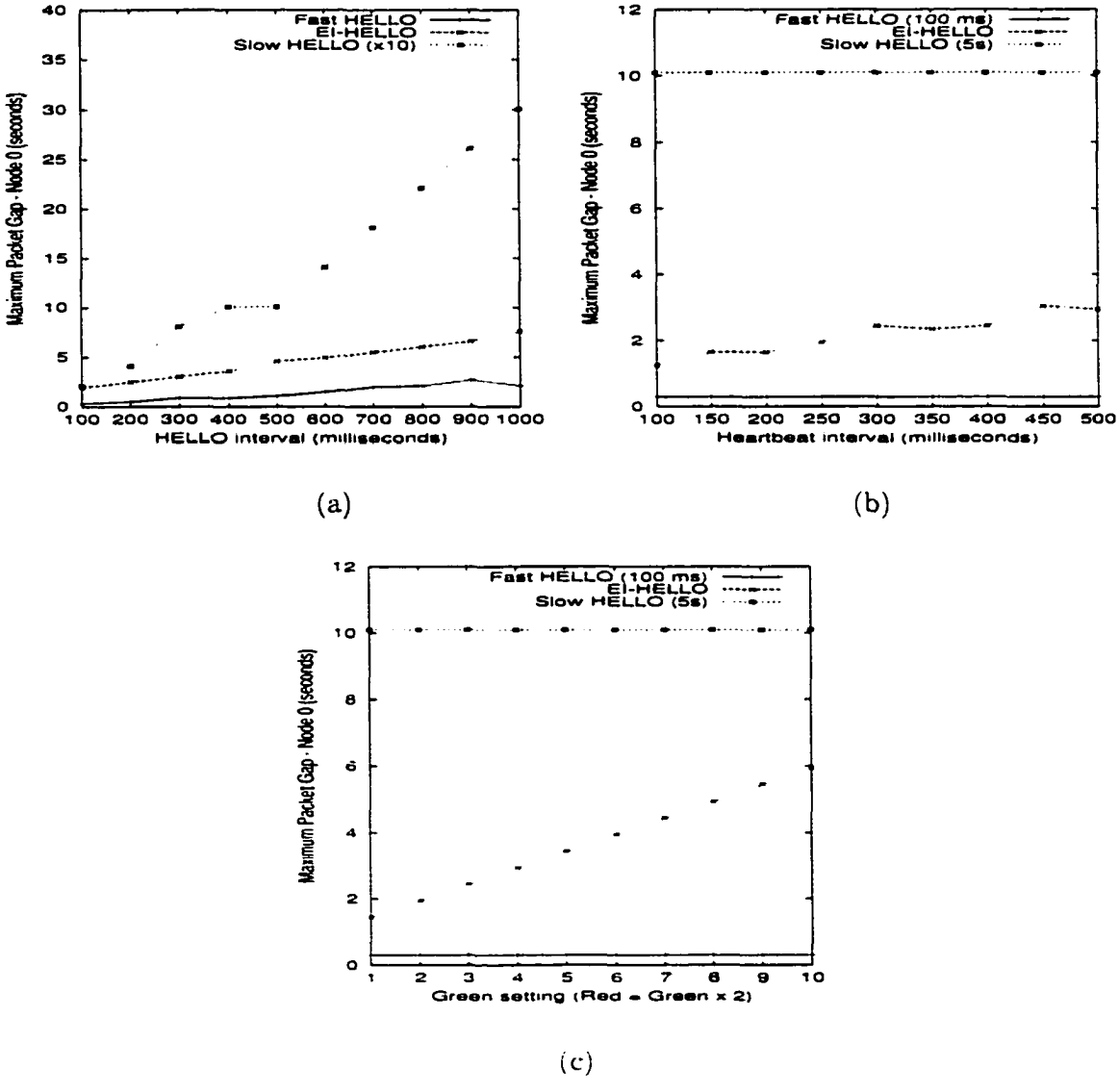


Figure 6.10 Effect of node failure - small network - (a) HELLO setting, (b) Heartbeat interval, (c) Green setting

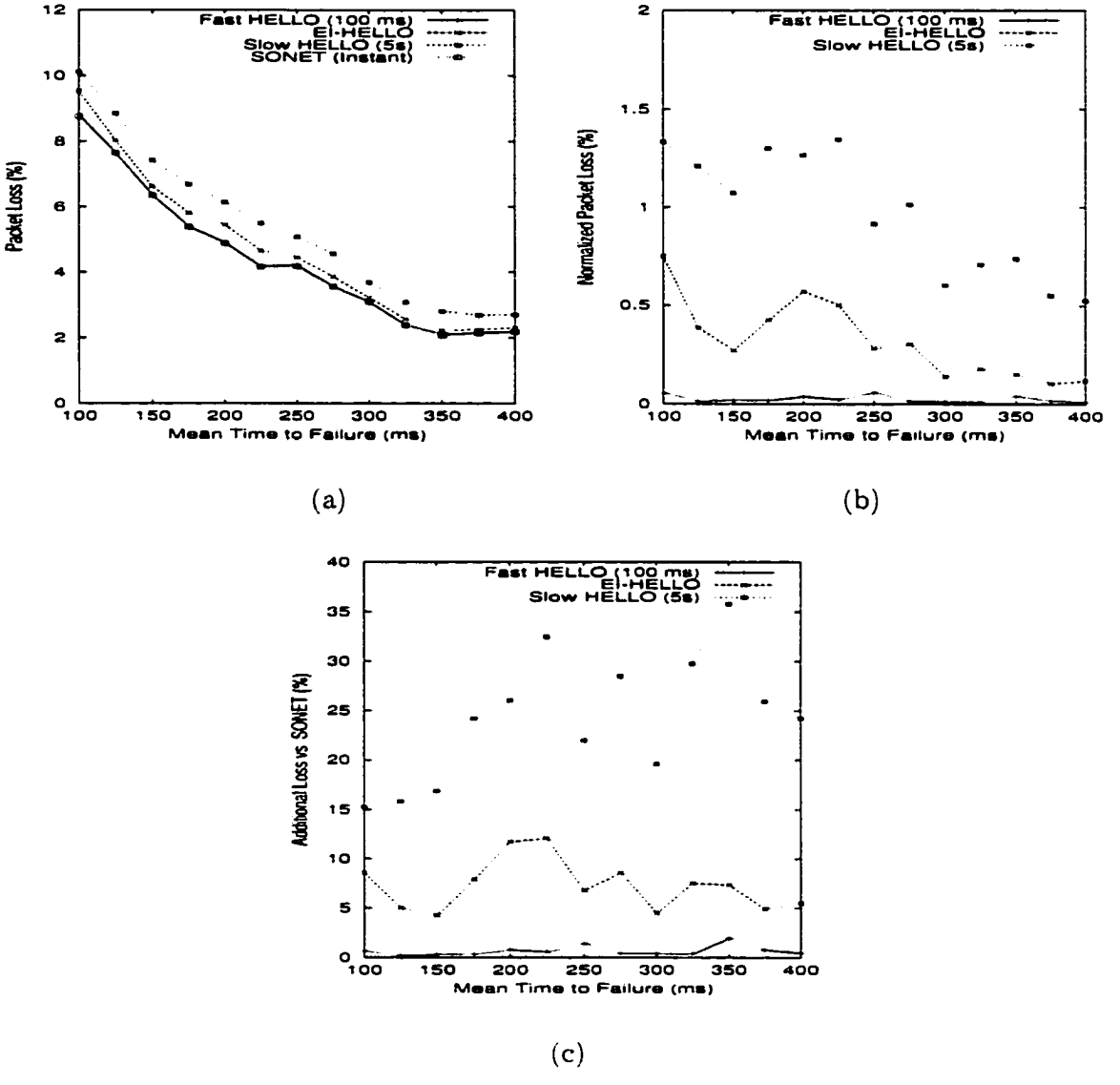


Figure 6.11 Effect of fault rate - NSFNet - (a) Raw packet loss. (b) Normalized (c) Normalized to SONET

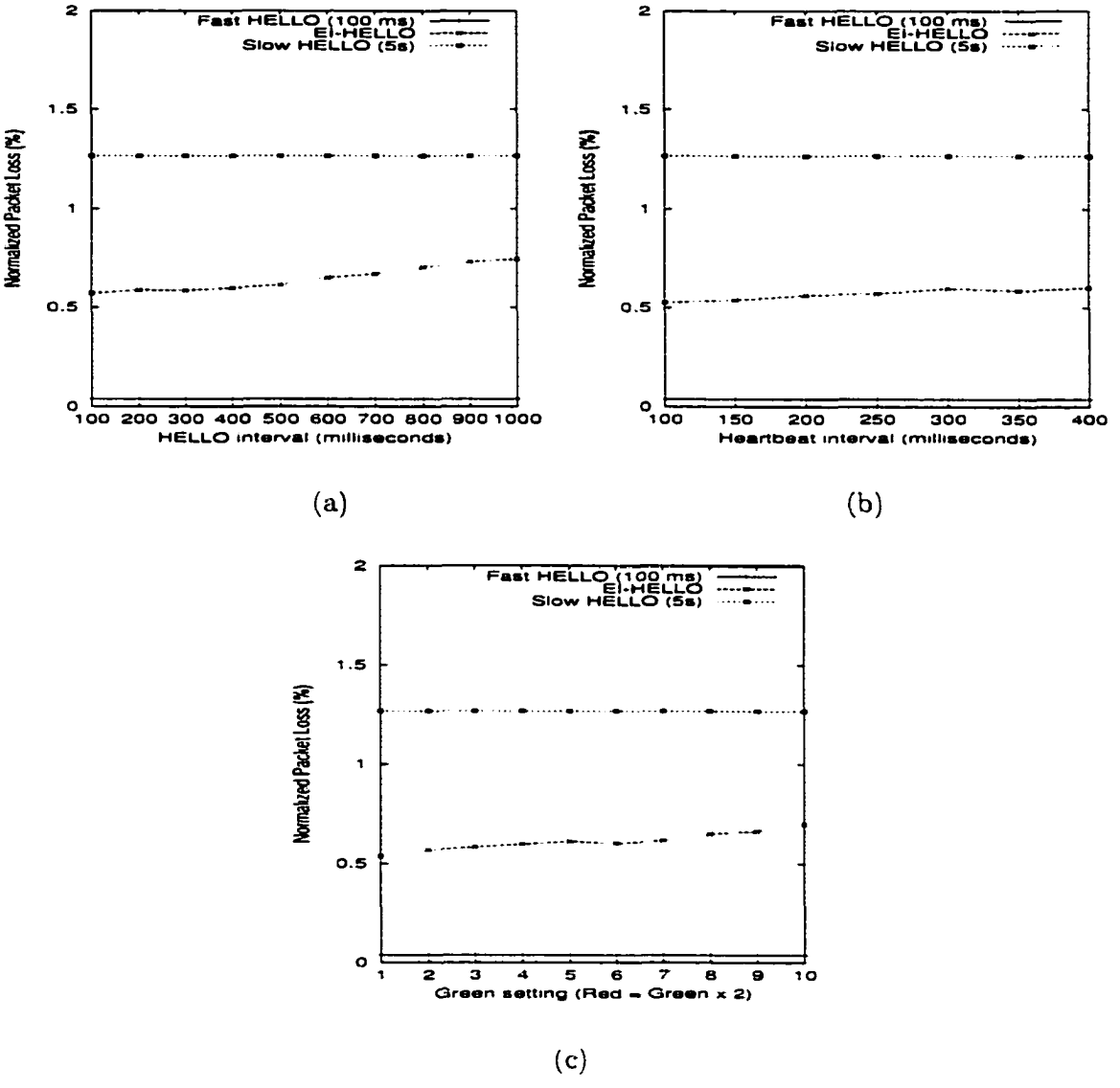


Figure 6.12 Effect of EI-HELLO parameters - NSFNet - (a) HELLO interval. (b) Heartbeat interval. (c) Green setting

timeout is increased as well, hence offering poorer performance. Out of all of the parameters, the heartbeat interval offers the best performance at an interval of 100 ms (see Figure 6.12(b)). However, this performance is only marginally better and comes at an extremely high message cost as examined in the next graph.

6.4.2.3 Effect of EI-HELLO Settings on Message Overhead

Figure 6.13 examines the effect of the heartbeat interval on the amount of message overhead introduced by the respective heartbeat and link state messages. As with Figure 6.12, the slow and fast HELLO models are kept at a constant setting for reference. The *green* setting and hybrid HELLO settings had minimal impact on message overhead since the hybrid HELLO is only invoked on faults and the *green* affects the trigger for such events. As the hybrid HELLO message setting has only a minimal impact on message overhead, it is reasonable to assume that the hybrid HELLO message could be lowered even further being constrained only by the link delay between the two nodes.

However, the heartbeat interval has a profound impact on the message overhead of the EI-HELLO model (see Figure 6.13). The profound impact of the heartbeat interval can be linked to several factors. First, each edge router follows a greedy paradigm whereby each router attempts to verify all of the routes to the other routers. Thus, for 17 edge routers, 17 packets are sent *HBI* times per second. At a rate of 100 ms, each node will receive 160 heartbeats packets per second. However, due to the fact that some multicast savings does occur (only 1 packet goes out from the node), the average overhead is slightly reduced. The edge-to-edge heartbeat quantity is amplified by the relative sparsity of NSFNet. Whereas in the small network example, several of the links did not require verification, many of the edge nodes in NSFNet send heartbeat packets along the same links. Hence, many of the packets end up simply re-verifying links that have been verified by other nodes due to the greediness of the EI-HELLO approach.

Thus, EI-HELLO suffers in a sparse network due to the unnecessary greediness of nodes. The performance could be improved by appropriately distributing the task of verification and

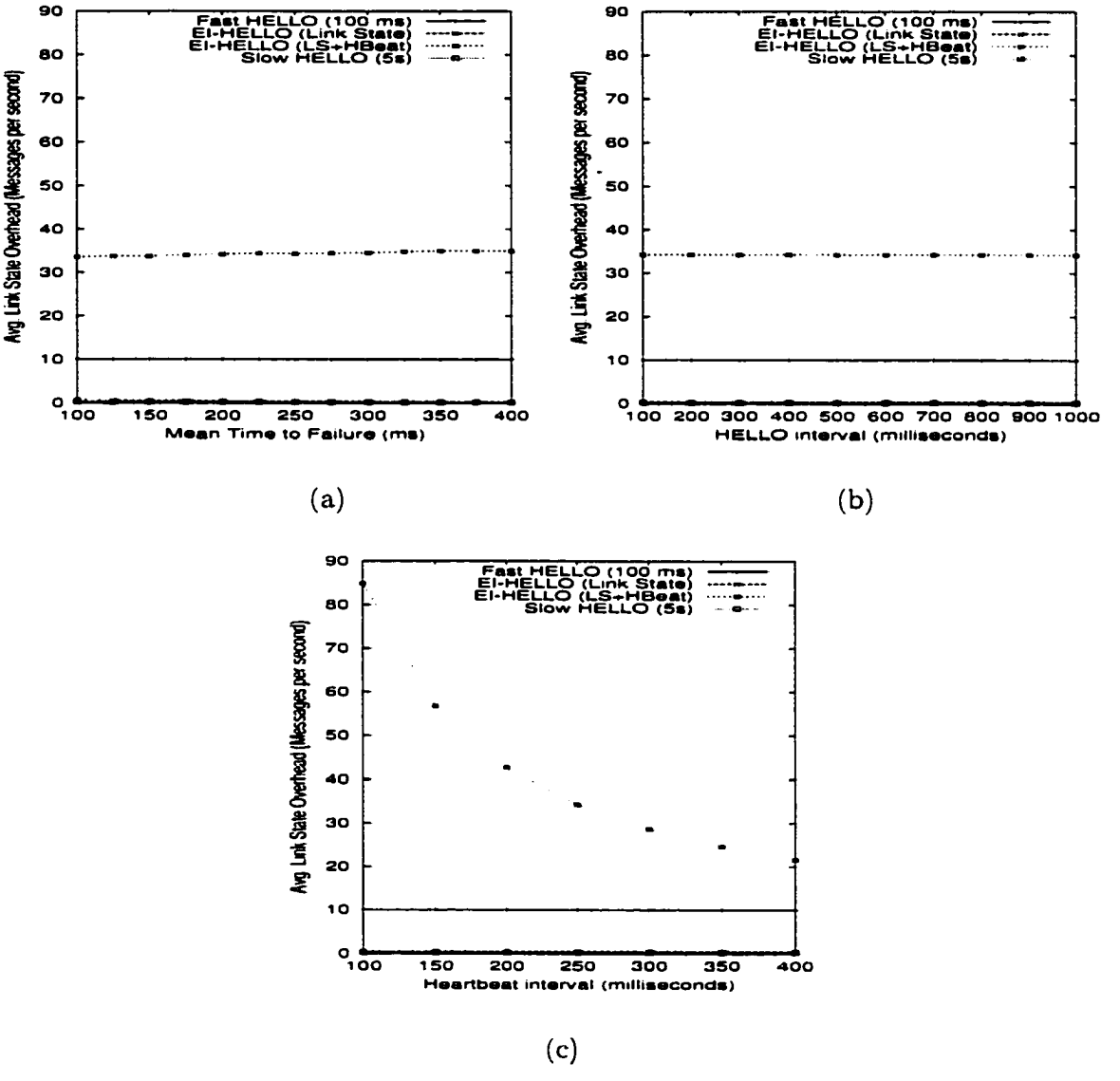


Figure 6.13 Effect on average message rate of (a) Fault rate, (b) HELLO interval, (c) Heartbeat interval

intelligently sharing the workload (see Section 6.3). However, such a greedy approach is not entirely poor as the heartbeat packets could also be used to gather core status information, thus distributing such information to the edge routers to make better routing decisions.

As a result, EI-HELLO is not ideal for all situations. If it is possible to reduce the HELLO interval to sub-second or have a Packet-SONET linkage, an approach such as EI-HELLO simply adds additional overhead. However, for cases where such a lower link state interval is not possible or where the gathering of core information is desired. EI-HELLO can offer significant benefit at a reasonable cost.

6.5 Conclusions

In this chapter, an approach (EI-HELLO) was proposed for edge-based health monitoring of core nodes in a DiffServ domain that captures the tradeoff of fault detection speed and link state overhead. The approach leveraged DSMCast for route-pinned heartbeat packets coupled with a hybrid HELLO mode to provide intelligent HELLO packet intervals.

The simulation studies showed that the EI-HELLO model could offer significant benefit over the traditional slow HELLO model currently in wide-scale deployment in the Internet. The improvement comes at the cost of additional bandwidth and does not offer significant gains over networks where sub-second HELLO intervals are possible. However, the implications of the EI-HELLO scheme for QoS monitoring of core routers can offer significant benefits beyond the initial purpose of fault detection. Thus, the EI-HELLO model offers significant promise and provides an excellent foundation for further research.

CHAPTER 7. CONCLUSIONS AND SUMMARY

In this chapter, a summary of the contributions of the dissertation is presented and additional considerations of the material are discussed. In addition, the chapter offers several concluding remarks regarding the use of the contributions and concludes with several future research directions.

7.1 Contributions

The work in this dissertation contributed several unique ideas to the field of networking, specifically the areas of Differentiated Services and multicasting. The notable contributions are summarized below:

- This work was the first to fully categorize all three conflicts between DiffServ and multicasting [100]. Most notably, the issue of scalability has been missing from the vast majority of existing works.
- The two architectures (DSMCast and EBM) were the first two architectures to satisfy the three primary conflicts between DiffServ and multicasting [73, 74, 75].
- The DSMCast architecture offered an architecture that could deliver core statelessness, heterogeneous QoS, and unified resource management with minimal packet overhead using hardware assistance.
- The bid/probe nature of DSMCast offered a unique approach for bridging the gap between sender versus receiver driven QoS in a DiffServ environment.

- The EBM architecture offered an alternative architecture that required zero modification to core routers but still offered the benefits of core statelessness, heterogeneous QoS, and unified resource management at the cost of additional bandwidth.
- The ECT algorithm uniquely captured the qualities of heterogeneous routing while balancing the tradeoff of tunnel length versus bandwidth consumption.
- The heterogeneous QoS work represented the first study of the effects of heterogeneous QoS in a single tree and the implications for users and service providers [76].
- The EI-HELLO model presented a unique approach for expediting fault detection by link state protocols and a future platform to explore for QoS management [77].
- The work was one of the first works to propose the use of dynamic QoS (variable PHBs) for not only multicast groups but also for unicast connections.

7.2 Additional Considerations

In addition to considerations of existing solutions discussed earlier in the two architectures, several other important areas also deserve mentioning. These areas include reliable multicast, filtered/layered multicast, as well as other security concerns.

7.2.1 Reliable Multicast

Although the original model for multicasting was based on the best-effort/unreliable delivery model of UDP, there has been considerable interest in developing a reliable multicast model [101, 102, 103, 104, 105]. Reliable multicast is especially applicable for areas such as content distribution networks (CDN). Both of the proposed architectures are compatible with reliable multicasting schemes but with several caveats.

First, a reliable multicast scheme implies that per-group monitoring is taking place in order to ensure reliable delivery. This support may range from caching of data for retransmission to simple NACK aggregation. Since neither DSMCast nor EBM allows for per-group state in the core of the network, the locations for reliable multicast entities would be restricted to

edge routers. Second, the performance of any reliable multicast scheme would suffer slightly. Since core routers cannot offer per-group intervention, support for operations even as simple as NACK aggregation are not possible at the core. As a result, the ingress router for a DiffServ domain may see significantly more NACKs than the router would see if core routers supported NACK aggregation. However, since aggregation still takes place at the edge routers of the domain in the first place, the impact of a NACKs to an ingress router is limited to the total number of egress points in a multicast group. In addition, the NACK would still be aggregated into a single NACK before reaching the sender, thus posing only a dilemma for the ingress router, not the source of the multicast group.

7.2.2 Heterogeneous QoS via RSVP Filters

Although the work in this dissertation focused on how to meet the heterogeneous QoS levels requested at different egress points of the domain, another facet of heterogeneous QoS is how the QoS request is conveyed to and mapped by the egress points themselves. One of the options for conveying such information is to use the filters of RSVP [26]. Under RSVP, receivers are able to allocate resources using either a *wildcard filter* (allow all senders), a *fixed filter* (allow a specific sender), or a *shared explicit filter* (allow certain senders). When two or more filters exist on a specific link, filters with the same specification may be merged together such that the highest request for bandwidth is satisfied. For instance, suppose that two receivers request a wildcard filter for the same multicast group with requested bandwidths of 2 Mb/s and 3 Mb/s. When the two filters come together on the tree, the 3 Mb/s bandwidth requirement is passed up the tree since it satisfies both the 2 Mb/s and the 3 Mb/s requirement.

One of the key differences with the two architectures versus RSVP filters is that both DSMCast and EBM are SSM-centric. As a result, there is no concept of a shared tree or multiple senders for a given group. Thus, only the fixed filter is truly required. However, both DSMCast and EBM do offer support for Anycasting to include legacy applications. In such a case, the filtering would only take place at the edge routers. As a result, several of the resource allocation mechanisms may not be as efficient since shaping must take place at the ingress

node and filtering/shaping will not take place in the core.

A possible solution is to use the variable QoS levels of DSMCast and EBM for not only heterogeneous in-tree support but also to use multiple distribution trees. For example, the 2 Mb/s and 3 Mb/s cases could use two trees, one giving the 2 Mb/s its requested QoS and the other tree using either a less than best effort PHB [37] or a drop PHB for traffic levels between 2 Mb/s and 3 Mb/s. The filtering may also be done as the packets arrive at the egress point to augment the PHB selection already done by the ingress router.

7.2.3 Security Concerns

A final area for concern lies in the realm of security. Although both architectures offer the opportunity for the addition of local domain-wise security, both approaches can offer potential new avenues for attack. Specifically, the route-pinned nature of both DSMCast and EBM (with the encapsulated option) can offer an opportunity for an attacker to disrupt the network.

For instance, an attacker could create a DSMCast packet that engages in triangle routing whereby the packet would continue to consume resources until the TTL was exhausted. Alternatively, the attacker could simply turn on all replication bits, thus causing significant replication until the TTL was exhausted. Without appropriate safeguards, an attacker could very quickly choke most of the bandwidth from the network with a relatively small number of packets.

However, these attacks represent an extreme case that can/have been addressed in several ways. First, an option is present in the DSMCast TEH options field to zero out the replication bits field, thus preventing triangle routing. The zeroing of the options field also defeats the second attack as well. Second, the DSMCast TEH can only be pushed into the packet at the edge of the domain. Thus, any packets arrive with a DSMCast TEH will be ignored as a new DSMCast TEH is always pushed onto multicast packets. Third, authentication and digital signatures can also be pushed into the header as well. A digital signature could be used to verify that the header originated from an edge router. Although compromising an edge router can foil these defenses, the entire DiffServ architecture fails if an edge router is compromised.

much more than just the multicasting portion. Finally, rate-limiting can also be pushed onto the core routers for multicast in general. Similar to how the EF class is rate-limited, all multicast traffic can be limited to a certain percentage of traffic, thus limiting the negative impact in the event of a misconfiguration or malicious attack.

Beyond a direct routing attacks, several other attacks can be applied to the architectures as well. For instance, a malicious attacker could send large packets to force fragmentation, thus consuming router CPU resources for the fragmentation. A similar attack might include rapid joining of multicast groups in the Anycasting scenario to spam significant floods and MSDP/MBGP traffic from the domain. Although these attacks do represent vulnerabilities, little can be done to prevent these attacks besides rate-limiting of high impact activities.

7.3 Conclusions

The work presented in this dissertation offers two architectures for providing multicast in a DiffServ environment. Although both architectures resolve the three conflicts between DiffServ and multicasting, each architecture has its respective strengths and weaknesses. As a result, each is better suited towards different network environments but both fit together towards offering efficient network support for multicasting.

7.3.1 EBM

For networks that do not currently support multicasting, the EBM architecture represents a relatively simple upgrade to offer complete multicasting support. To incorporate the EBM architecture, a router needs only to offer software support at the edge routers of the domain. In fact, the majority of complexity for multicasting (tree construction, protocol mapping/translation) can be migrated to the Multicast Broker (MB), thus making multicast support at the edge routers even easier. An edge router need only recognize new join/leave requests, tunnel/replicate multicast packets, and keep group membership information. If the edge routers have sufficient computational resources, the tasks of the MB can be migrated to each edge router, provided that SSM is employed for all join/leave requests. A hybrid ap-

proach can easily be employed as well with the MB handling legacy Anycast requests and edge (ingress) routers handling SSM requests.

7.3.2 DSMCast

The next step for multicast support would be to move from EBM to a DSMCast architecture. Since EBM already abstracts the multicast transport functionality, DSMCast is simply replacing the how the packets themselves are transported. DSMCast offers a better overall QoS and bandwidth efficiency over EBM but comes at the cost of requiring hardware assistance at each of the core routers. As an intermediate step or on networks with low levels of multicast traffic, the processing of the DSMCast TEH could be done via software although such an approach is merely an interim step. Since DSMCast involves simply comparing a specific pattern in the multicast header, the majority of commercially available network processors or FPGAs could be programmed to support DSMCast.

While the architectures do overcome the primary issue of core scalability, both architectures still require a significant effort for deployment on the part of the ISP. However, this effort can be minimized provided that either architecture is deployed at the same time DiffServ functionality is also deployed. By tying both DiffServ and multicasting together, an ISP can deliver two compelling services at the same time, thus offering significant benefits over a unicast-only best-effort network.

7.3.3 Heterogeneous QoS

The work on heterogeneous QoS offered interesting insights into how heterogeneous QoS performs in a single multicast tree. Most notably, the tradeoff of network efficiency versus differentiation is a difficult problem that is an open topic for research. While optimizing for bandwidth is goal for an ISP, the economic issues are more likely to outweigh the technical issues in this case.

7.3.4 Fault Detection

Finally, the work on fault detection represents an interesting premise to explore future work. Although the problem of fault detection may be easily solved by simply deploying L2-L3 linkages (similar to Packet-over-SONET [96]) , the solution is still valid for the majority of the Internet [94]. In addition, this approach has significant implications beyond assisting with the fault detection of the link state protocol.

First, the work offer promise in regards to QoS monitoring of the core of the network. Rather than having to use a reactionary scheme, the health/QoS of the core of the network is provided continually and can be acted upon in a proactive fashion. Several existing projects such as MINC (Multicast-based Inference of Network-internal Characteristics) have examined this on a large scale using multicasting [106]. However, the route-pinned nature of DSMCast allows DSMCast to act as an efficient SNMP (Simple Network Management Protocol) with significantly less overhead. In addition, the fault tolerance work has applicability in other disciplines as well such as Processing in Memory [107].

7.4 Future Research Directions

Finally, although this dissertation has answered many of the initial problems posed by DiffServ multicasting, the work has opened up several other interesting areas that merit future research. These areas include:

- *Dynamic unicast QoS:* The issue of variable PHBs for unicast connections is an unexplored area in DiffServ. With a fixed PHB from edge-to-edge, a service provider is able to offer only absolute levels of QoS to users. In practice, it may be desirable to offer gradients of QoS to users. By embedding dynamic PHB information in the packet, the PHB may change as the packet traverses the DS domain. The implications of such research can dramatically broaden the QoS levels offered by DiffServ. However, the cost of such a scheme is not free and thus, further research into the cost-benefit tradeoff analysis of such a scheme is needed.

- *Dynamic routing:* Since both architectures offer the option for carrying the replication/routing information inside the packet, it is possible to dynamically route the packet according to the QoS of the network. Whereas traditional IP multicast would require all of the nodes to be aware for tree rearrangement, the entire tree can be rearranged and updated only at the ingress router. The use of dynamic routing could be used to offer better network utilization or to offer dynamic tree rearrangement in response to QoS [88, 108].

- *Feedback-based QoS adaptation:* A second issue to be explored is the issue of feedback-based QoS adaptation for DiffServ. Although feedback-based QoS adaptation has been mentioned in the literature [87], it has not been thoroughly explored in DiffServ multicasting. In addition, the concept of using network dynamics from core routers to determine packet markings at the edge routers rather than edge-to-edge feedback is also an unexplored issue. The QoS information provided by DSMCast and EI-HELLO offer excellent mechanisms for gathering such information. Feedback-based QoS adaptation in DiffServ has significant implications for relative DiffServ as well as for any dynamic QoS scheme.

- *Negative QoS:* A unique concept introduced by the use of single trees for heterogeneous QoS is the use of negative QoS/negative shaping at the edge router. In addition to the QoS introduced by the network, packets could be negatively shaped (additional delay or additional loss) in order to balance out the Good Neighbor Effect. This approach runs contrary to most QoS schemes whereby the goal is to offer better overall QoS, not worsening of the QoS. Such an approach balances the need for network efficiency with the need for QoS differentiation. However, the issues of determining how to negatively shape a packet are subjects for future research.

ACKNOWLEDGEMENTS

I would like to thank the following people for their thoughts and time regarding my various papers and dissertation:

Brian Carpenter of IBM-Zurich, Christopher Metz of Cisco Systems, Tony Speakman of Cisco Systems, Jennifer Rexford of AT&T, Abdelmadjid Bouabdallah of Universite de Techologie-Compiegne, Hatem Bettahar of Universite de Techologie-Compiegne, Raphi Rom of Technion - Israel Institute of Technology , Michael Speer of Sun Microsystems, Christoph Schuba of Sun Microsystems, Anirban Chakrabarti of Iowa State University, and my advisor, Manimaran Govindarasu of Iowa State University.

BIBLIOGRAPHY

[1] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks." in *Proc. of ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, Nov. 2002.

[2] J. Kangasharju, K. W. Ross, and J. Roberts, "Performance Evaluation of Redirection Schemes in Content Distribution Networks," in *Proc. of WCW 2000*, 2000.

[3] "The Ins and Outs of Content Delivery Networks," *Stardust.com White Paper*, Dec. 2000.

[4] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m,k)-firm guarantees." *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1443–1451, Dec. 1995.

[5] A. Striegel and G. Manimaran, "Best-effort scheduling of (m,k)-firm real-time streams in multihop networks," *Computer Communications*, vol. 23, no. 13, pp. 1292–1300, July 2000.

[6] R. West, K. Schwan, and C. Poellabauer. "Scalable Scheduling Support for Loss and Delay Constrained Media Streams." in *Proc. IEEE Real-time Technology and Application Symposium (RTAS)*, 1999.

[7] C. Siva Ram Murthy and G. Manimaran. *Resource Management in Real-time Systems and Networks*, MIT Press, Apr. 2001.

[8] R. Atkinson, "IP Authentication Header." *IETF RFC 1826*, Aug. 1995.

[9] S. Kent and R. Atkinson. "IP Encapsulating Security Payload (ESP)," *IETF RFC 2406*, Nov. 1998.

[10] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding PHB (per-hop behavior)," *IETF RFC 3246*, Mar. 2002.

[11] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," *IETF RFC 2597*, June 1999.

[12] N. Christin, J. Liebeherr, and T. Abdelzaher, "A quantitative assured forwarding service," in *Proc. of IEEE INFOCOM*, New York, NY, June 2002.

[13] J. Liebeherr and N. Christin, "Rate allocation and buffer management for Differentiated Services," *Computer Networks*, vol. 40, no. 1, Aug. 2002.

[14] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," in *Proc. of ACM SIGCOMM*, 1999, pp. 109–120.

[15] A. Striegel and G. Manimaran, "Packet scheduling with delay and loss differentiation," *Computer Communications*, vol. 25, no. 1, pp. 21–31, Jan. 2002.

[16] A. Pras, B. van Beijnum, R. Sprenkels, and R. Parhonyi, "Internet accounting," *IEEE Communications Magazine*, vol. 39, no. 5, May 2001.

[17] G. Carle, F. Hartanto, M. Smirnov, and T. Zseby, "Charging and accounting for qos-enhanced ip multicast," in *Proc. of Protocols for High-Speed Networks (PfHSN)*, Salem, MA, Aug. 1999.

[18] X. Wang and H. Schulzrinne, "Pricing network resources for adaptive applications in a Differentiated Services network," in *Proc. of IEEE INFOCOM*, Anchorage, Alaska, Apr. 2001.

[19] A. M. Odlyzko, ." *Journal of Computer Resource Management*, pp. 23–27, Spring 2001.

[20] D. S. Milojicic et al., "Peer-to-peer computing," *HP Labs Technical Report*, Mar. 2002.

[21] J. Postel, "Internet protocol," *IETF RFC 791*, Sept. 1981.

[22] J. Postel, "Service mappings," *IETF RFC 795*, Sept. 1981.

[23] R.Braden, D. Clark, and S. Shenkar, "Integrated Services in the Internet architecture: An overview," *IETF RFC 1633*, June 1994.

[24] S. Shenkar, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," *IETF RFC 2212*, Sept. 1997.

[25] J. Wroclawski, "Specification of the controlled-load network element service." *IETF RFC 2211*, Sept. 1997.

[26] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSeVation Protocol (RSVP) Version 1, Functional Specification," *IETF RFC 2205*, Sept. 1997.

[27] K. Nichols, S. Blake, F. Baker, and D.L. Black, "Definition of the Differentiated Services field (DS Field) in the IPv4 and IPv6 headers," *IETF RFC 2474*, Dec. 1998.

[28] S. Blake et. al. "An Architecture for Differentiated Services." *IETF RFC 2475*, Dec. 1998.

[29] F. Reichmeyer, K. Chan, D. Durham, R. Yavatkar, S. Gai, K. McCloghrie, and S. Herzog, "COPS usage for differentiated services." *IETF Internet Draft*, Nov. 1998, Work in Progress.

[30] I. Khalil and T. Braun, "A Range Based SLA and Edge Driven Virtual Core Provisioning in Diffserv-VPNs," in *Proc. of IEEE Local Computer Networks (LCN)*, Tampa, Florida, Nov. 2001.

[31] H. Schulzrinne, H. Tschofenig, X. Fu, J. Eisl, and R. Hancock, "CASP - cross-application signaling protocol." *IETF Internet Draft draft-schulzrinne-nsis-casp-00.txt*, Sept. 2002, Work in Progress.

[32] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP." *IETF RFC 2481*, Jan. 1999.

[33] A.K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 344–357, June 1993.

[34] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, Aug. 1995.

[35] J.C.R. Bennett and H. Zhang, "Hierarchical Packet Fair Queuing Algorithms." *IEEE/ACM Transactions on Networking*, vol. 5, pp. 675–689. Oct. 1997.

[36] M.K.H. Leung and J.C.S. Lui and D.K.Y. Yau, "Adaptive Proportional Delay Differentiated Services: Characterization and Performance Evaluation." *IEEE/ACM Transactions on Networking*, vol. 9. no. 6. Dec. 2001.

[37] R. Bless and K. Wehrle, "A lower than best-effort per-hop behavior." *IETF Internet Draft draft-bless-diffserv-lbe-phb-00.txt*. Aug. 1999, Work in Progress.

[38] "Building Next Generationg Network Processors," *Agere Systems White Paper*. Apr. 2001.

[39] C-Y. Lee and N. Seddigh, "Controlling the number of egress points in dynamic multicast groups." *IETF Internet Draft draft-leecy-multicast-egress-limit-00.txt*, Oct. 1999, Work in Progress.

[40] R. Bless and K. Wehrle, "IP Multicast in Differentiated Services Networks." *IETF Internet Draft draft-bless-diffserv-multicast-02.txt*, Nov. 2001. Work in Progress.

[41] B. Yang and P. Mohapatra, "Multicasting in Differentiated Service domains." in *Proc. of IEEE GLOBECOM*, 2002.

[42] F. Hao, E. Zegura, and M. Ammar, "QoS Routing for Anycast Communications: Motivation and an Architecture for DiffServ Networks," *IEEE Communications*, pp. 48–56, June 2002.

[43] "IETF NSIS (Next Step in Internet Signaling) working group," Oct. 2002. http://www.ietf.org/html.charters/nsis-charter.html.

[44] A. Striegel and G. Manimaran, "Dynamic class-based queue managements for scalable media servers," *Journal of Systems and Software*, 2002. to appear.

[45] Y. Xu and R. Guerin, "Individual qos versus aggregate qos: A loss performance study," June 2002.

[46] A. Habib, S. Fahmy, and B. Bhargava, "Design and evaluation of an adaptive traffic conditioner for differentiated services networks," in *Proc. of IEEE International Conference on Computer Communications and Networks (IC3N)*, Phoenix, Arizona. Oct. 2001, pp. 90–95.

[47] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment," *IEEE Network*, vol. 14, pp. 10–20, Jan./Feb. 2000.

[48] P. Van Mieghem and G. Hooghiemstra and R. van der Hofstad. "On the Effiency of Multicast," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, Dec. 2001.

[49] R.C. Chalmers and K. Almeroth. "Modeling the branching characteristics and efficiency gains in global multicast trees." in *Proc. of IEEE INFOCOM*. Apr. 2001, pp. 449–458.

[50] D.R. Cheriton and S. Deering, "Host groups: A multicast extension for datagram internetworks," in *Proc. Data Communications Symposium*, 1985, pp. 172–179.

[51] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for IP multicast service and architecture." *IEEE Network*, pp. 78–89. Jan./Feb. 2000.

[52] B. Wang and J. C. Hou, "Multicast Routing and its QoS Extension." *IEEE Network*, pp. 22–36, Jan./Feb. 2000.

[53] M. Ramalho, "Intra- and Inter- domain multicast routing protocols: A survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 1, pp. 2–25, Jan.-Mar. 2000.

[66] R. Bless and K. Wehrle, "Group Communication in Differentiated Services Networks." in *Internet QoS for the Global Computing 2001 (IQ 2001), Workshop at CCGRID 2001*, Brisbane, Australia, May 2001, pp. 618–625.

[67] G. Bianchi, N. Blefari-Melazzi, G. Bonafede, and E. Tintinelli, "QUASIMODO: QUAlity of ServIce-aware Multicasting Over Diffserv and Overlay networks," *IEEE Network*. 2003. To appear.

[68] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification," *IETF RFC 2362*, June 1998.

[69] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Protocol Specification (Revised)," *IETF Internet Draft draft-ietf-pim-sm-v2-new-05.txt*, Mar. 2002, Work in Progress.

[70] G. Bianchi and N. Blefari-Melazzi, "Admission Control over Assured Forwarding PHBs: A Way to Provide Service Accuracy in a DiffServ Framework." in *Proc. of GLOBECOM*, San Antonio, Texas, Nov. 2001.

[71] Z. Li and P. Mohapatra, "QoS-Aware Multicasting in DiffServ Domains." in *Proc. of Global Internet Symposium*. 2002.

[72] I. Stoica, T.S. Ng, and H. Zhang, "REUNITE: A Recursive Unicast Approach for Multicast," in *Proc. of IEEE INFOCOM*, 1999.

[73] A. Striegel and G. Manimaran, "A scalable approach to DiffServ multicasting," in *Proc. of ICC'2001*, Helsinki, Finland, June 2001.

[74] A. Striegel and G. Manimaran, "A scalable protocol for member join/leave in DiffServ multicast," in *Proc. of Local Computer Networks (LCN)*, Tampa, Florida, Nov. 2001.

[75] A. Striegel, A. Bouabdallah, H. Bettahar, and G. Manimaran, "EBM: A new approach for scalable DiffServ multicasting," in *IEEE INFOCOM*, 2003, Submitted.

[76] A. Striegel and G. Manimaran, "Dynamic DSCPs for heterogeneous QoS in DiffServ multicasting," in *Proc. of GLOBECOM*, Taipei, Taiwan, Nov. 2002.

[77] A. Striegel and G. Manimaran, "Edge-based fault-detection in DiffServ networks," in *Proc. of IEEE Intl. Conference on Dependable Systems and Networks (DSN)*, Washington, D.C., June 2002.

[78] "Network processing forum," Oct. 2002, http://www.npforum.org/.

[79] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner, "Scalable high speed IP routing lookups," in *Proceedings of SIGCOMM '97*, Sept. 1997, pp. 25–36.

[80] J. Moy, "OSPF Version 2," *IETF RFC 1583*, June 1994.

[81] International Standards Organization. "Intra-Domain IS-IS Routing Protocol." *ISO/IEC JTCI/SC6 WG2 N323*. Sept. 1989.

[82] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[83] A. Adams, J. Nicholas, and W. Siadak, "Protocol independent multicast - dense mode (PIM-DM): Protocol specification (revised)," *IETF Internet Draft draft-ietf-pim-dm-new-v2-01.txt*, Feb. 2002, Work in Progress.

[84] "Multicast Source Discovery Protocol," *Cisco IP Multicast Training Materials*, 2000.

[85] "Multi-protocol BGP," *Cisco IP Multicast Training Materials*, 2000.

[86] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol extensions for BGP-4." *IETF RFC 2283*, Feb. 1998.

[87] C. Dovrolis and P. Ramanathan, "A case for relative differentiated services and the proportional differentiation model," *IEEE Network*, pp. 26–34, Sept.-Oct. 1999.

[88] R. Sriram, G. Manimaran, and C. Siva Ram Murthy, "A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees," *IEEE/ACM Trans. Networking*, pp. 514–529, Aug. 1999.

[89] R. Boivie, "A New Multicast Scheme for Small Groups," *IBM Research Report RC21512*, June 1999.

[90] "UCB/LBNL/VINT Network Simulator - ns (version 2)," Available at www.mash.cs.berkeley.edu/ns/.

[91] A. Striegel, "GenMCast: A generic multicast extension for ns-2." *ND CSE Technical Report*, To be published.

[92] "NSFNet T3 Backbone Service," *Merit Network Inc.*, Oct. 1992.

[93] T. Ballardie, P. Francis, and J. Crowcroft, "Core-based trees (CBT): An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, 1993, pp. 85–95.

[94] C. Alaettinoglu, V. Jacobson, and H. Yu, "Toward Millisecond IGP Convergence," in *Proc. of NANOG 20*, Washington, D.C., Oct. 2000.

[95] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," *IETF RFC 1771*, Mar. 1995.

[96] K. Ravishankar and J. Koshy, "Packet over SONET interface: A design strategy," *WIPRO Technologies White Paper*, Dec. 2001.

[97] J. Parker, D. McPherson, and C. Alaettinoglu, "Short Adjacency Hold Times in IS-IS," *IETF Internet Draft draft-parker-short-isis-hold-times-01.txt*, July 2001, Work in progress.

[98] E. Rosen, A. Visiwanathan, and R. Callon, "Multiprotocol label switching architecture," *IETF RFC 3031*, Jan. 2001.

[99] M. Maekawa, A. Oldehoft, and R. Oldehoft, *Operating Systems Advanced Concepts*, The Benjamin Cummings Publishing Company, Inc., Menlo Park, California, 1987.

[100] A. Striegel and G. Manimaran, "A survey of qos multicasting issues," *IEEE Communications*, pp. 82–87, June 2002.

[101] B. Levine and J. J. Garcia-Luna-Aceves, "A comparison of reliable multicast protocols," *Multimedia Systems*, vol. 6, no. 5, pp. 334–348, 1998.

[102] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 784–803, Dec. 1997.

[103] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (RMTP)," *IEEE Journal of Selected Areas in Communications*, vol. 15, no. 3, pp. 407–421, 1997.

[104] K. Yano and S. McCanne, "A window-based congestion control for reliable multicast based on TCP dynamics," in *ACM Multimedia*, 2000, pp. 249–258.

[105] T. Speakman et. al, "PGM reliable transport protocol specification," *RFC 3208*, Dec. 2001.

[106] R. Caceres, N.G. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2462–2480, Nov. 1999.

[107] P. Kogge, J. Brockman, and V. Freeh, "Processing-in-memory based systems: Performance evaluation considerations," in *In Workshop on Performance Analysis and its Impact on Design, PAID'98, held in conjunction with the International Symposium on Computer Architecture*, Barcelona, Spain, June 1998.

[108] F. Bauer and A. Varma, "ARIES: A Rearrangeable Inexpensive Edge-Based Steiner Algorithm," *IEEE JSAC*, vol. 15, no. 3, pp. 382–397, Apr. 1997.